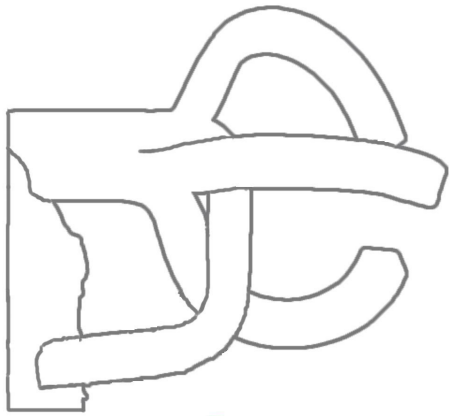


XIV Jornadas de Ingeniería del Software y Bases de Datos

San Sebastián, 8-11 de septiembre de 2009

Editores:
Antonio Vallecillo
Dept. Lenguajes y Ciencias de la Computación
ETSI Informática
Universidad de Málaga
Bulevar Louis Pasteur, 35. 29071 Málaga. Spain
e-mail: av@lcc.uma.es

Goiuria Sagardui
Departamento de Electrónica e Informática
Escuela Politécnica Superior
Mondragon Unibertsitatea
Loramendi, 4; 20500 Arrasate-Mondragón. Spain
e-mail: gsagardui@eps.mondragon.edu

Filmación e impresión: Gráficas Michelena

Depósito Legal: SS-988-2009

ISBN: 978-84-692-4211-7

Prólogo

Las XIV Jornadas de Ingeniería del Software y Bases de Datos se celebraron en San Sebastián del 8 al 11 de septiembre de 2009, en el incomparable marco del Palacio Miramar, siendo organizadas por la Escuela Politécnica Superior de la Universidad de Mondragón. Como viene ocurriendo desde la edición del 2001, se han celebrado, en paralelo y compartiendo algunos actos, de las IX Jornadas de Programación y Lenguajes (PROLE). Ambos eventos son organizados bajo los auspicios de SISTEDES, la Sociedad de Ingeniería del Software y Tecnologías de Desarrollo de Software.

Este volumen recoge los trabajos seleccionados por el Comité de Programa de JISBD 2009. Este año se han incluido tres tipos distintos de contribuciones en las actas. En primer lugar tenemos los artículos de investigación originales, que describen resultados de investigación o experiencias industriales relativas a los campos de la Ingeniería del Software y de las Bases de Datos. Se recibieron un total de 81 resumenes preliminares, de los cuales 75 se plasmaron finalmente en artículos. Entre ellos el Comité de Programa decidió seleccionar 22 como artículos largos. Esto ha supuesto un ratio de aceptación del 29 %, lo que demuestra el arduo proceso de revisión y selección al que fueron sometidos los artículos, así como la calidad de los finalmente seleccionados. Además de estos artículos, otros 5 fueron seleccionados para participar en la conferencia como artículos cortos, con la idea de favorecer y estimular el debate científico entre los asistentes y dar cabida a la presentación de trabajos incipientes. Todos los artículos fueron revisados siguiendo un sistema de revisión por pares, y discutidos entre los miembros del Comité.

Además de este tipo de artículos, este año también se ha incluido en las actas los resúmenes de las demostraciones de herramientas presentadas, y de los artículos relevantes ya publicados. Las demostraciones de herramientas son el camino para demostrar la viabilidad práctica de las propuestas teóricas y metodológicas formuladas por los equipos de investigación, y una muestra de lo que pueden aportar a la ciencia y a la industria. En esta ocasión se seleccionaron 23 herramientas para ser brevemente presentadas durante las sesiones, y expuestas durante las Jornadas.

Por otro lado, la madurez de la comunidad JISBD se plasma en un número creciente de trabajos publicados en revistas y congresos de primera línea. Muchos de estos trabajos pasan desapercibidos para la comunidad al no tener un reflejo en las propias Jornadas, más orientadas hacia trabajos emergentes. Por ello, JISBD acoge desde las últimas dos ediciones la presentación de este tipo de publicaciones, con un doble objetivo: por una parte, publicitarlos dentro de las propias jornadas; por otra, ofrecerlos como guía y estímulo al resto de la comunidad. En esta ocasión se incluyen en las actas los resúmenes de los 10 trabajos seleccionados por la organización en esta categoría.

El éxito de la conferencia JISBD también se refleja en el número de eventos que suceden a su alrededor. En primer lugar, JISBD 2009 contó con tres conferenciantes invitados de primer nivel: Don Batory (Department of Computer Sciences, University of Texas at Austin, EE.UU.) que impartió la charla “Stepwise Development of Streaming Software Architectures”; Jean Bézivin (University of Nantes, Francia) con “Advances

in Model Driven Engineering"; y Houari Sahraoui (Universidad de Montreal, Canada), cuya charla fue "Put the Horse before the Cart: Task-Driven Development of Software Visualization Tools". Nuestro agradecimiento más sincero por su disponibilidad para aceptar la invitación y venir a San Sebatián a impartir sus conferencias.

El programa de JISBD 2009 también incluyó dos tutoriales sobre temas de candente actualidad, como son las líneas de producto y el desarrollo dirigido por modelos.

Otro de los puntos fuertes de JISBD son los Talleres, que conforman punto de reunión obligado para los investigadores que trabajan en algunos temas de especial interés, y que ofrecen un foro de discusión excelente para estimular el debate y la colaboración entre ellos. Este año se ha contado con 6 Talleres, algunos de amplia tradición como ADIS (9ª edición), DSDM (6ª edición) o PRIS (4ª edición) y otros más noveles como son PNIS, ZOCO o WASELF.

Nos gustaría expresar nuestro más sincero agradecimiento a todos los miembros del Comité de Programa por su tiempo y dedicación a la hora de revisar y seleccionar los artículos que fueron finalmente aceptados para su presentación, y que han permitido confeccionar un año más un programa de gran calidad y nivel. Por supuesto, queremos también agradecer a los autores que enviaron artículos a las Jornadas, fueran finalmente aceptados o no, por el esfuerzo realizado y por su contribución al evento.

También queremos agradecer desde aquí al equipo organizador todo su esfuerzo y trabajo. Esto incluye a los organizadores locales de la Universidad de Mondragón, que han permitido hacer realidad esta conferencia, así como a los distintos Coordinadores que se han ocupado de organizar aspectos esenciales como los Talleres (Coral Calero), Demostraciones (Juan de Lara), Tutoriales (Ernest Teniente), Divulgación de Trabajos Relevantes ya Publicados (Belén Vela), Publicidad (Gentzane Aldekoa y José Raúl Romero), Actas (Leire Etxeberria) y Web (Ana Altuna). Nos gustaría también mostrar nuestro agradecimiento al Comité Permanente de las JISBD: primero, por depositar en nosotros su confianza a la hora de presidir el Comité de Programa y organizar la conferencia; y segundo, por su constante apoyo y soporte. Mención especial merece Oscar Díaz, cuyos consejos y ayuda han sido siempre inestimables. También mencionar el sistema de revisión de artículos que hemos utilizado, EasyChair, que fue de gran utilidad y ayuda durante todo el proceso de revisión y para la confección de estas actas.

Nuestro agradecimiento explícito a los patrocinadores del evento que hicieron posible que la conferencia fuera todo un éxito: las empresas Intersystems y Ulma, la Asociación de Técnicos en Informática (ATI), la revista Novática, las Universidades de Mondragón y del País Vasco, Cursos de Verano, el Gobierno Vasco, el Ministerio de Ciencia e Innovación, la caja de ahorros Caja Laboral y la Corporación Mondragón.

Muchas gracias a todos los asistentes a las JISBD 2009, y esperamos verles de nuevo en las próximas JISBD 2010.

San Sebastián, Septiembre 2009

Antonio Vallecillo
Goiuria Sagardui

Comité Ejecutivo

Presidente del Comité de Programa
Antonio Vallecillo (Univ. Málaga)

Presidenta del Comité Organizador
Goiuria Sagardui (Univ. Mondragón)

Coordinadora de Talleres
Coral Calero (Univ. Castilla-La Mancha)

Coordinador de Demostraciones
Juan de Lara (Univ. Autónoma Madrid)

Coordinador de Tutoriales
Ernest Teniente (Univ. Polit. Cataluña)

Coordinadora de Divulgación de Trabajos Relevantes ya Publicados
Belén Vela (Univ. Rey Juan Carlos)

Coordinadores de Publicidad
Gentzane Aldekoa (Univ. Mondragón)
José Raúl Romero (Univ. Córdoba)

Coordinadora de actas
Leire Etxeberria (Univ. Mondragón)

Coordinadora de la Web
Ana Altuna (Univ. Mondragón)

Comité Organizador (Univ. Mondragón)

Goiuria Sagardui
Joseba Andoni Aguirre
Gentzane Aldekoa
Ana Altuna
Merixell Armentia
Lorea Belategui
Itxaso Buruaga
Leire Etxeberria
Osane Lizarralde
Urtzi Markiegi
Xabier Sagarna

Comité de Programa

Abrahão, Silvia (Univ. Polit. Valencia)
Acuña, Cesar (Univ. Rey Juan Carlos)
Aldana, José (Univ. Málaga)
Álvarez, Bárbara (Univ. Polit. Cartagena)
Aramburu, María José (Univ. Jaume I)
Araujo, Joao (Univ. Nova Lisboa)
Ávila, Orlando (Open Canarias)
Bañares, Jose A. (Univ. Zaragoza)
Berlanga, Rafael (Univ. Jaume I)
Bertrand, Enrique (Software AG)
Boronat, Artur (Univ. Leicester)
Brisaboa, Nieves (Univ. Coruña)
Cabot, Jordi (Univ. Oberta Cataluña)
Cachero, Cristina (Univ. Alicante)
Canal, Carlos (Univ. Málaga)
Canos, José Hilario (Univ. Polit. Valencia)
Carretero, Baltasar (T-Systems)
Cavero, José María (Univ. Rey Juan Carlos)
Corchuelo, Rafael (Univ. Sevilla)
Costal, Dolors (Univ. Polit. Cataluña)
Crespo, Yania (Univ. Valladolid)
de la Riva, Claudio (Univ. Oviedo)
Delgado Kloos, Carlos (Univ. Carlos III)
Dolado, Javier (Univ. País Vasco)
Duran, Francisco (Univ. Málaga)
Fdez-Bertoa, Manuel (Univ. Málaga)
Fdez-Medina, Eduardo (Univ. Castilla-La Mancha)
Franch, Xavier (Univ. Polit. Cataluña)
Fredlund, Lars-Ake (Univ. Polit. Madrid)
Fuente, Pablo de la (Univ. Valladolid)
Garbajosa, Juan (Univ. Polit. Madrid)
García Molina, Jesús (Univ. Murcia)
García, Félix (Univ. Castilla-La Mancha)
Gaspar da Silva, Mario (Univ. Lisboa)
Genero, Marcela (Univ. Castilla-La Mancha)
Genova, Gonzalo (Univ. Carlos III)
Gómez, Jaime (Univ. Alicante)
Gonzalez, Daniel (Univ. La Laguna)
Goñi, Alfredo (Univ. País Vasco)
Guerra, Esther (Univ. Carlos III)
Hernández, Juan (Univ. Extremadura)
Hogdson, Peter (Procedimientos Uno)
Irastorza, Arantza (Univ. País Vasco)
Iribarne, Luis (Univ. Almeria)
Iturrioz, Jon (Univ. País Vasco)
Juristo, Natalia (Univ. Polit. Madrid)
Laguna, Miguel A. (Univ. Valladolid)
Llorens, Juan (Univ. Carlos III)
Lopes, Antonia (Univ. Lisboa)
Lopez Cobo, Jose M.(XimetriX)
Lozano, Adolfo (Univ. Extremadura)
Melia, Santiago (Univ. Alicante)
Mena, Eduardo (Univ. Zaragoza)
Moreira, Ana (Univ. Nova Lisboa)
Moreno, Ana María (Univ. Polit. Madrid)
Pelechano, Vicente (Univ. Polit. Valencia)
Pimentel, Ernesto (Univ. Málaga)
Polo, Antonio (Univ. Extremadura)
Quer, Carme (Univ. Polit. Cataluña)
Riquelme, José (Univ. Sevilla)
Rito, Antonio (Univ. Tec. Lisboa)
Roda, José Luis (Univ. La Laguna)
Ruíz, Francisco (Univ. Castilla-La Mancha)
Ruíz-Cortés, Antonio (Univ. Sevilla)
Sagardui, Goiuria (Univ. Mondragón)
Samos, José (Univ. Granada)
Sánchez, Fernando (Univ. Extremadura)
Sánchez, Juan (Univ. Polit. Valencia)
Sánchez, Victor (Open Canarias)
Toro, Miguel (Univ. Sevilla)
Toval, Ambrosio (Univ. Murcia)
Trujillo, Juan Carlos (Univ. Alicante)
Trujillo, Salvador (Ikerlan)
Tuya, Javier (Univ. Oviedo)
Urpí, Toni (Univ. Polit. Cataluña)
Vicente, Cristina (Univ. Polit. Cartagena)

Revisores

Miguel Ángel Martínez
Luz Marina Moreno de Antonio
Isabel Brito
Nuno Cardoso
Dante Carrizo
Ana Cerdeira-Pena
Antonio Corral
Diego Alonso
Javier Cámara
Jose María García
Irene Garrigós
Anna Grimán
Jorge García
Ignacio García
Sergio Ilarri
Ernesto Jimenez
Beatriz Bernárdez
Joaquín Lasheras
Manuel Llavador
Esperanza Manso
Miguel Angel Martinez-Prieto
Enric Mayol
Jose-Norberto Mazon
Fernando Molina
Sonia Montagud
Mª Ángeles Moraga
Isabel A. Nepomuceno-Chamorro
Jesús Pardillo
Juan Ángel Pastor
Jennifer Perez
Javier Pérez
Manuel Resinas
Roberto Rodríguez-Echeverría
Roberto Ruiz
Sergio Segura
Manuel Serrano
Pedro Sánchez
Mari Carmen Otero
Jose Zubcoff

Patrocinadores

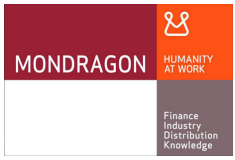


Tabla de contenidos

IV Sesión 3. Requisitos / Ingeniería del software empírica

V Sesión 4. MDE y Transformaciones

VI Sesión 5. BBDDs y Tecnologías de SGBD

VII Sesión 6. Recuperación de Información, Indexación y BD en Web

VIII Sesión 7. Líneas de Producto

IX Sesión 8. Ontologías, Web semántica

X Sesión 9. Validación/Modelado Conceptual

XI Sesión 10. Calidad, Medición y Estimación de Productos y Procesos Software

XII Talleres

XIII Tutoriales

Parte I

Conferencias Invitadas

Advances in Model Driven Engineering

Jean Bézivin

University of Nantes

Abstract. When the MDA^{TM} was first introduced, the main idea was to produce a platform specific model (e.g., a Java program) from a platform independent model (e.g., a UML model). As more general Model Driven Engineering (MDE) practices met success, models increasingly represent new situations or phenomena not initially imagined. In addition to development-time models, maintenance-time and execution-time models are more and more frequently used. But we have now a richer classification of models. For example we may have models of the data or models of the dynamic behavior of a system and much more. This has much impact on the operations that may apply to these different kinds of models. The talk will summarize the past evolution and discuss the current research agenda in MDE based on the more precise model classification and broader spectrum of operation on them. The quest for unification of software artifacts with models is still going on but the applicability scope of MDE is at the same time rapidly broadening. The synergy between these two processes produces a very active research and development field that will be discussed in the presentation.

Short Biography

Jean Bezivin is professor of Computer Science at the University of Nantes, France He presently leads at Ecole des Mines de Nantes the AtlanMod INRIA research team. He got a Master degree from the University of Grenoble and a Ph.D. from the University of Rennes. Since 1980 he has been very active in Europe in the object-oriented community, starting the ECOOP series of conference, the TOOLS series of conferences, and more recently the MODELS/<<UML>> and the ICMT series of conferences. He founded in 1979, at the University of Nantes, one of the first Master programs in Software Engineering entirely devoted to Object Technology (Data Bases, Concurrency, Languages and Programming, Analysis and Design, etc.). His present research interests include model engineering and more especially the techniques of model transformation applied to data engineering and to software forward and reverse engineering. He has published many papers and organized tutorials and workshops in the domains of concurrency, simulation, object-oriented programming, and model-driven engineering. On the subjects of model driven engineering and MDA^{TM}, he has been leading the OFTA industrial group in France, co-animating a CNRS specific action and the Dagstuhl seminar #04101. He is a member of the ECOOP, MODELS, TOOLS and ICMT steering committees. He was co-chair of ECOOP'2006 and PC chair of TOOLS'2007

Actas JISBD 2009, pp.3, ISBN: 978-84-692-4211-7

Stepwise Development of Streaming Software Architectures ⋆

Don Batory

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas, 78712 U.S.A.

Abstract. We present simple and practical Model Driven Engineering techniques to develop, explain, test, and verify designs of streaming software architectures. We develop component-connector architectural models incrementally by transforming them into progressively more elaborate architectures, and optimize these architectures for efficiency and crash-fault tolerance. Two case studies are presented: join parallelizations in database machines and recoverable, crash fault-tolerant server architectures.

Short Biography

Don Batory holds the David Bruton Centennial Professorship in the Department of Computer Sciences at The University of Texas at Austin. He was an Associate Editor of IEEE Transactions on Software Engineering (1999-2002), Associate Editor of ACM Transactions on Database Systems (1986-1992), a member of the ACM Software Systems Award Committee (1989-1993; Committee Chairman in 1992), Program Co-Chair for the 2002 Generative Programming and Component Engineering Conference. He is a leading researcher on feature-oriented software development. Over the last ten years, he and his students have written 11 Award Papers for their work in automated and component-based program development.

⋆ Joint work with Taylor L. Riche

Actas JISBD 2009, pp.4, ISBN: 978-84-692-4211-7

Put the Horse before the Cart: Task-Driven Development of Software Visualization Tools

Houari Sahraoui

Universidad de Montreal, Canada

Abstract. Software visualization is an efficient and flexible tool to perform maintenance tasks that are difficult to automate. It allows inspection and analysis of large sets of software data at various levels of detail. Many visualization environments have been proposed during the last two decades. For the most, they use interesting visualization metaphors and are powerful and scalable. On the other hand, one cannot but notice that they are almost unused outside the small community that developed them. One explanation of these phenomena is that the adequacy of the environments for the targeted tasks is questionable. Indeed, the choices made during the development of these environments are not (explicitly) motivated by the nature of the targeted tasks. This makes it difficult for common users to understand, learn, and use them in concrete situations. In this talk, we present principles and guidelines to develop software visualization tools by considering explicitly the nature of the data and actions involved in maintenance tasks. These principles are illustrated through the visualization environment VERSO.

Short Biography

Houari A. Sahraoui is Professor of Software Engineering in the Department of Computer Science and Operations Research of the University of Montreal. He obtained his Ph.D. degree in 1995 in computer science, with specialization in meta-modeling and model transformation, from the Pierre & Marie Curie University, Paris. His research interests include the application of artificial-intelligence techniques to software engineering, software visualization, object-oriented measurement and quality, and reengineering. He has been on the program, steering, or organization committees of many international, IEEE and ACM conferences, and is member of the editorial board of three journals. He was the general chair of the IEEE Automated Software Engineering conference in 2003.

Actas JISBD 2009, pp.5, ISBN: 978-84-692-4211-7

Parte II

Sesión 1. Verificación y pruebas

Generación de Pruebas Basada en Restricciones para Consultas SQL

María José Suárez-Cabal, Claudio de la Riva, Javier Tuya

Departamento de Informática – Universidad de Oviedo
Campus Universitario de Gijón, s/n
{cabal,claudio,tuya}@uniovi.es

Abstract. En este artículo se describe un enfoque para generar automáticamente datos de prueba para consultas SQL, incorporando tanto información del esquema de la base de datos como de las propias consultas a probar. Las diferentes situaciones de prueba sobre las consultas se identifican a partir de un criterio de cobertura de condiciones y se representan mediante un conjunto de restricciones que la base de datos debe cumplir. Utilizando el *constraint solver* Alloy, se generan instancias de la base de datos que satisfacen las restricciones, y que constituyen una carga de prueba de la base datos. Se ilustra y evalúa la propuesta con un caso de estudio de una aplicación real.

Keywords: Pruebas del Software, Criterio MCDC, Pruebas sobre Base de Datos, Generación de datos de prueba, Alloy *constraint solver*

1 Introducción

Entre las actividades más comúnmente utilizadas en la prueba del software destacan la generación y ejecución de los casos de prueba. Mientras que la ejecución de los casos de prueba es, en muchos casos, fácilmente automatizable, la generación de los datos de prueba es, en la mayoría de los casos, una actividad con un fuerte componente manual, y por lo tanto, muy sujeta a errores. Esta generación manual es especialmente dificultosa en aquellos casos donde las entradas son estructuras de datos complejas, como ocurre en el caso de aplicaciones basadas en bases de datos.

Varios trabajos y herramientas han contemplado el problema relativo a la generación de instancias de base de datos, como por ejemplo [5] [2] [10]. Sin embargo, en la mayoría de las situaciones, los procesos de generación de datos tienen en cuenta únicamente la estructura de la base de datos y sus restricciones, pero no las características de las consultas a probar, por lo que los datos generados pueden no ser útiles para la prueba de las consultas.

Recientemente, la problemática asociada a las pruebas sobre bases de datos ha centrado la atención de varios investigadores, desde diferentes puntos de vista. Uno de ellos es la definición de criterios de suficiencia que permitan evaluar la calidad de los datos de prueba en relación con las consultas SQL y que a la vez sirvan como guía para el diseño de nuevos casos de prueba [8] [10] [11]. Otro es el relativo a la generación de los datos de prueba tomando como base la información de la consulta a

probar [1] [15] [4]. En esta segunda categoría, algunos autores [16] [9], proponen el uso de *contraints solvers* como artefacto para la generación de los datos de prueba, modelando el esquema y ciertas partes de la consulta como un conjunto de restricciones, y resolviendo posteriormente dichas restricciones con un *constraint solver*. Sin embargo, en general, los enfoques utilizados no tienen en cuenta la compleja relación estructural de las tablas del esquema y la consulta SQL ni se especifican claramente los criterios utilizados para la generación de los casos, por lo que no consideran las características intrínsecas propias del lenguaje SQL, como por ejemplo el manejo de valores nulos o el soporte para clausulas join.

En este trabajo se combinan los dos enfoques anteriores. Por una parte se utiliza un criterio de cobertura específico para el lenguaje SQL [12] para identificar las diferentes situaciones de prueba y por otro se generan de forma automática los datos de prueba contemplando dichas situaciones con un enfoque basado en restricciones. Para la generación de dicha pruebas se utiliza el *constraint solver* Alloy, que permite, por una parte, especificar relaciones estructurales entre conjuntos y las restricciones entre ellos (*Alloy Language*), y por otra, analizar de forma automática la especificación (*Alloy Analyzer*). Para ello, las diferentes situaciones de prueba se representan en el lenguaje como un conjunto de restricciones que se resuelven posteriormente con el analizador con el objetivo de generar una base de datos de prueba que satisfaga las situaciones de prueba para la consulta SQL. El enfoque soporta la definición de esquemas de la base de datos con restricciones de claves primarias, claves ajenas, valores nulos y valores de dominios, y de consultas SQL que incluyen joins internas y externas, así como predicados en la cláusula WHERE. Además, permite la generación de datos de prueba para varias consultas SQL de forma simultánea. La aplicación sobre un caso de estudio muestra la viabilidad de la propuesta.

El resto de artículo es como sigue. En la sección 2 se presentan los conceptos básicos relativos al criterio de cobertura utilizado y al *constraint solver* Alloy. En la sección 3 se incluye el enfoque propuesto por los autores para la generación de las pruebas y la sección 4 muestra la aplicación a un caso de estudio. El artículo finaliza en la sección 5 con conclusiones y trabajo futuro.

2 Conceptos Previos

En esta sección se describe brevemente el criterio de cobertura utilizado y el lenguaje y el analizador Alloy que se utilizarán para la generación de los datos de prueba.

2.1 Criterio de Cobertura para Consultas SQL

En [12] se describe un criterio de cobertura que adapta el criterio de *Multiple Condicion/Decisión Modificado* (MCDC) [3] para el diseño datos de prueba para consultas SQL. Este criterio, denominado SQLFpc, es automatizado y mejorado en [14] y consiste en definir un conjunto de requisitos de prueba (*reglas de cobertura*) que especifican diferentes situaciones de los datos de prueba que se deben de cubrir

con el objetivo de probar la consulta. A continuación se muestra un ejemplo que ilustra el criterio.

Considérese la siguiente consulta SQL:

```
SELECT * FROM Profesor P INNER JOIN AsignaturaImpartida A ON
P.pid=A.pref WHERE A.creditos <> 2 OR A.responsable=1
```

Esta consulta selecciona los profesores y las asignaturas que imparten, donde el número de créditos impartidos en la asignatura es distinto de 2 o el profesor que la imparte es responsable (puede ser 1 ó 0, indicando si es o no responsable, respectivamente). La consulta se ejecuta sobre una base de datos que contiene información sobre los profesores (tabla `Profesor`, donde `pid` es la clave primaria), asignaturas (tabla `Asignatura` donde `aid` es la clave primaria) y el detalle de las asignaturas que imparten los profesores (tabla `AsignaturaImpartida` donde `aiid` es la clave primaria y los atributos `pref` y `aref` son claves ajenas que hacen referencia a `Profesor` y `Asignatura` respectivamente). Además, los profesores y asignaturas tienen un nombre y en la tabla `AsignaturaImpartida` el atributo `creditos` puede tener valor nulo, pero no el atributo `responsable`.

Para la identificación de las diferentes situaciones a cubrir en la prueba, el criterio utilizado tiene en cuenta, entre otras, las condiciones relativas de la cláusula WHERE así como las relativas a los operadores JOIN de la consulta. Así para la consulta anterior, se deberían diseñar datos de prueba que cubran las siguientes situaciones:

- Incluir filas en las tablas `Profesor` y `AsignaturaImpartida` donde las condiciones del WHERE sean: (C1) ambas falsas, (C2) cierto y falso y (C3) falso y cierto.
- Como el número de créditos puede ser nulo, incluir filas tales que (N1) el número de créditos impartidos sea nulo y la condición (`A.responsable=1`) sea falsa.
- Debido a la presencia de un operador JOIN: (J1) incluir al menos un profesor que no imparta asignaturas y (J2) incluir al menos una fila en la tabla `AsignaturaImpartida` que no tenga profesores relacionados y que la condición del WHERE sea cierta (nótese que esta situación es posible ya que `pref` puede ser nulo).

Cada una de las situaciones anteriores se expresa mediante una sentencia SQL que se denomina *regla de cobertura*. A modo de ejemplo, se muestran las reglas de cobertura para las situaciones (C2) y (J1) anteriores:

```
(C2): SELECT * FROM Profesor P INNER JOIN AsignaturaImpartida A ON
P.pid=A.pref WHERE (A.creditos <> 2) AND NOT(A.responsable = 1)
(J1): SELECT * FROM Profesor P LEFT JOIN AsignaturaImpartida A ON
P.pid=A.pref WHERE (A.pref IS NULL) AND (P.pid IS NOT NULL)
```

Dada una base de datos cargada con datos de prueba, una determinada situación de prueba es cubierta si la ejecución de la regla de cobertura sobre dicha base de datos genera al menos una fila como salida.

2.2 El Lenguaje y el Analizador Alloy

Alloy [6] es un lenguaje de modelado estructural basado en lógica de primer orden que es adecuado para expresar relaciones estructurales complejas y restricciones entre

los diferentes elementos, y cuyos modelos pueden ser analizados utilizando el Analizador de Alloy (*Alloy Analizer*) [7].

Un modelo en Alloy consiste en la definición de un conjunto de átomos o elementos (`sig`) y la definición de un conjunto de restricciones (`fact` y `pred`) sobre los elementos.

Como ejemplo, considérese la siguiente especificación en Alloy que representa a la tabla `Profesor` del ejemplo de la sección anterior:

```
sig varchar {}
one sig Profesor { filas: Int -> varchar }
fact{ all x: filas.varchar | x > 0 }
```

La primea definición `sig` modela el tipo de dato `varchar` para el atributo que representa al nombre de un profesor (`pnombre`). Esta definición es necesaria ya que Alloy únicamente soporta como tipo de dato básico el conjunto de enteros (`Int`). La segunda declaración `sig` representa la tabla `Profesor` (el cuantificador `one` indica que únicamente existirá una tabla `Profesor`). Ésta se define como una relación `filas` de `Int` a `varchar` que modela las tuplas (`filas`) de la tabla `Profesor` (el operador '`->`' representa el producto cartesiano entre conjuntos). El dominio de la relación representa al atributo identificador del profesor (`pid`) y la imagen modela al atributo `pnombre`. Así los elementos de la relación `filas` son tuplas de la forma <`pid`,`pnombre`>.

Las restricciones `fact` denotan restricciones que siempre se deben cumplir en el modelo. En el ejemplo, indica que todo elemento perteneciente al dominio de la relación `filas` (`all x: filas.varchar`), esto es, los elementos del atributo `pid`, deben ser mayores que cero (`x>0`).

Además, en Alloy se pueden especificar otras restricciones en forma de predicados (`pred`). Un predicado define una restricción parametrizada que puede ser invocada desde cualquier punto de la especificación instanciando sus parámetros y devuelve un valor cierto o falso. Análogamente, también se pueden definir funciones (`fun`) que especifican expresiones (también parametrizadas) que devuelven un valor. Por ejemplo, la siguiente especificación define un predicado `test` para la tabla `Profesor` que denota la existencia de al menos una fila (cuantificador `some`) con valores del atributo `pid` entre `2` y `7` (la función `pid` devuelve el domino del atributo `pid` de `Profesor`):

```
fun pid [t:Profesor]:Int{ (t.filas).varchar }
pred test [t:Profesor]{
    some x: pid [t] | x >= 2 and x <= 7
}
```

La ejecución del analizador teniendo en cuenta el predicado anterior (`run test`) produce, por ejemplo, la siguiente instancia:

```
Profesor.filas={<3,varchar>,<1,varchar>,<1,varchar>}
```

En este caso se instruye al analizador para buscar instancias que satisfagan tanto el predicado `test` (`some x: pid[t]| x>=2 and x<=7`) como las restricciones asociadas en la definición de la tabla (`all x:filas.varchar|x>0`), esto es, el atributo `pid` debe siempre mayor que 0 y existirán algunas filas donde su valor está comprendido entre 2 y 7, ambos inclusive.

3 Generación de Datos de Prueba

Dado un esquema de una base de datos y un conjunto de reglas de cobertura de una consulta SQL, el enfoque propuesto en este artículo para la generación de datos de prueba es el siguiente:

1. Representar en Alloy cada una de las tablas (y sus restricciones) contenidas en el esquema de la base de datos
2. Representar en Alloy cada una de las reglas de cobertura como restricciones que se deben cumplir en el modelo obtenido en el punto anterior.
3. Ejecutar el analizador de Alloy con el objetivo de encontrar una instancia de la base de datos (conjunto de filas en cada una de las tablas) que satisfaga todas las restricciones

   A continuación se describen cada uno de estos puntos.

3.1 Representación del Esquema de la Base de Datos

Relaciones (Tablas). Dado un conjunto de atributos $A_1,\ldots,A_n$ con dominios $D_1,\ldots,D_n$, respectivamente, una relación $R(A_1{:}D_1, \ldots, A_n{:}D_n)$ o bien $R(A)$ es un subconjunto del producto cartesiano $D_1x...xD_n$. Cada una de las tablas pertenecientes al esquema, se modela en Alloy de acuerdo al siguiente patrón:

```
//Definición de tipos de los dominios (distintos de Int)
sig D1 {}
…
sig Dn {}
one sig NULL {} //Modela el valor NULL
// Definición de la tabla
one sig R {
    filas: (D1+NULL)->…->(Dn+NULL)
}
```

Cada tabla en el esquema se define mediante un tipo con una relación (`filas`) sobre el producto cartesiano de los dominios de los atributos. Así, cada uno de los elementos de la relación es una tupla $<a_1,\ldots, a_n> \in D_1x\ldots xD_n$.

Una característica fundamental del modelo relacional es la presencia de información desconocida en los valores de los atributos (valores nulos, `NULL`). Para representar y posteriormente manejar este valor, se define un nuevo tipo con un único elemento (`one sig NULL {}`). Dado que puede estar presente en cualquiera de los atributos, se extiende cada uno de los dominios con este valor (`Di+NULL`), donde '+' es el operador de unión de conjuntos en Alloy.

Restricción de Valores Nulos. Esta restricción (NOT NULL) impide que un determinado atributo tome valores nulos. La especificación en Alloy de esta restricción para el atributo A_1 de la relación $R(A)$ es la siguiente:

```
pred isNotNULL [x:D1+NULL] {x in D1}
pred isNULL [x:D1+NULL] {x in NULL}
fact notNULL{  all x:filas.(D2+NULL).….(Dn+NULL)| isNotNULL [x]}
```

Esta restricción indica que cada elemento del domino correspondiente al atributo A_1 (`all x: filas.(D2+NULL).….(Dn+NULL)`) no toma valores nulos. El

predicado `isNotNULL[x]` devuelve cierto si `x` pertenece al conjunto de enteros (el operador `in` es el operador de pertenencia de conjuntos en Alloy). Análogamente, `isNULL[x]` devuelve cierto si el valor de `x` es NULL.

Restricción de Clave Primaria. En el modelo relacional, la restricción de clave primaria (PRIMARY KEY) indica que cada fila de una relación está identificada de forma única por un atributo o conjunto de atributos, que conforman la clave. A continuación se muestra la especificación en Alloy para la restricción de clave primaria del atributo A_1 en la relación R(A):

```
fact primaryKey{ all x: filas.(D2+NULL).….(Dn+NULL)| one x.rfilas }
```

La restricción `primaryKey` expresa que cada elemento del dominio correspondiente al atributo A_1 se corresponde exactamente con una tupla en la relación `filas` (`one x.filas`).

Restricción de Clave Ajena. La restricción de clave ajena (FOREING KEY) identifica un atributo o conjunto de atributos en una relación R que referencia a una tupla en otra relación S. Dadas dos relaciones R(A) y S(B) donde B_1 es la clave primaria de S(B), la siguiente especificación en Alloy define la restricción de clave ajena para el atributo A_n en R(A), asumiendo que su valor puede ser nulo:

```
fun pk [s:S]: B1+NULL {s.filas.(D2+NULL).….(Dm+NULL)}
fact foreingKey{
    let An = (Dm-1+NULL).….(D2+NULL).filas) | An in (pk[S] + NULL)}
```

La función `pk[S]` devuelve el dominio correspondiente a la clave primaria de la relación S. La restricción `foreingKey` expresa que los elementos correspondientes al atributo A_n de R(A) (A_n=`(Dm-1+NULL).….(D1+NULL).filas`) son un subconjunto de los correspondiente a la clave primaria de S(B) (A_n `in (pk[S]+NULL)`).

Como ejemplo, la Fig. 1 muestra la especificación en Alloy para la tabla `AsignaturaImpartida` del ejemplo de la sección 2.1.

```
one sig NULL {}
//Funciones y predicados de uso general
pred isNotNULL [x:Int+NULL]{x in Int}
pred isNULL [x:Int+NULL]{x in NULL}
//Definición: AsignaturaImpartida:(aiid,creditos,responsable,pref,aref)
one sig AsignaturaImpartida {
    filas: (Int+NULL)->(Int+NULL)->(Int+NULL)->(Int+NULL)->(Int +NULL)
}{  //Restricciones
    all x:aiid[filas]|one x.filas  // Clave primaria PK
    all x:aiid[filas]|isNotNULL[x] // Valor no nulo de PK
    all x:responsable[filas]|isNotNULL[x] //Valor no nulo para respons.
    let fk1=pref[filas]| fk1 in pk[Profesor]+NULL //FK para profesor
    let fk2=aref[filas]| fk2 in pk[Asignatura]+NULL //FK para asignatura
}
//Funciones de acceso a los atributos de la tabla
fun aiid [t:AsignaturaImpartida.filas]: Int + NULL {
   t.(Int+NULL).(Int+NULL).(Int+NULL).(Int+NULL)  }
fun creditos [t:AsignaturaImpartida.filas]: Int + NULL {
   (Int+NULL).(t.(Int+NULL).(Int+NULL).(Int+NULL))  } […]
```

Fig. 1. Especificación Alloy para la tabla AsignaturaImpartida

3.2 Representación de las Reglas de Cobertura

Una regla de cobertura es una expresión relacional formada por operadores SELECT y/o operadores JOIN. Como notación general para las reglas se usará la expresión $R[p(A,B)]^J S[q(A,B)]$, donde R(A) y S(B) son las relaciones que participan en la consulta, p(A,B) es la condición del JOIN (cláusula ON), J identifica el tipo de JOIN (I-Inner, L-Left, R-Right) y q[A,B] es el predicado de la cláusula WHERE. Por ejemplo, la expresión $P[pid=pref]^I A[(creditos<>2)$ AND NOT(responsable=1)] denotaría a la regla C2 del ejemplo de la sección 2.1:

```
(C2): SELECT * FROM Profesor P INNER JOIN AsignaturaImpartida A ON
P.pid = A.pref WHERE (A.creditos <> 2) AND NOT(A.responsable = 1)
```

Dado que cada regla de cobertura expresa una situación de prueba para la cual se han de generar datos de prueba, cada una de ellas se especificará en Alloy como un predicado. A continuación, se muestra la especificación en Alloy de las reglas de cobertura que contienen JOINs y cláusulas WHERE (para reglas sin JOIN y con JOIN encadenados la representación sería similar).

Reglas de cobertura con INNER JOIN. Indican una situación de prueba donde deben existir filas en R(A) y S(B) relacionadas a través del predicado p(A,B) y que cumplan el predicado q(A,B):

```
some r: R.filas,s:S.filas|p(A[r],B[s])and q(A[r],B[s])
```

El predicado expresa la restricción que al menos debe existir alguna tupla en R(A) y S(B) (`some r: R.filas, s:S.filas`) tal que se cumpla al mismo tiempo el predicado de la cláusula JOIN y el predicado de la cláusula WHERE (`p(A[r],B[s])and q(A[r],B[s])`).

Reglas de cobertura con LEFT JOIN. Expresan una situación de prueba donde exista al menos una fila en R(A) que no tenga ninguna relacionada en S(B):

```
some r:R.filas | A[r] not in B[S.filas]
```

Reglas de cobertura con RIGHT JOIN. Análoga a la anterior, estas reglas representan una situación de prueba donde exista al menos una fila en la relación S(B) que no tenga ninguna relacionada en la relación R(A):

```
some s:S.filas| B[s] not in A[R.filas]
```

Como ejemplo, considérese la regla C2 anterior. Esta regla se representaría con el siguiente predicado:

```
some p:Profesor.filas, a:AsignaturaImpartida.filas |
    (pid[p]=pref[a]) and (creditos[a]!=2 ] and (responsable[a]!=1))
```

Sin embargo, la evaluación de este predicado podría producir resultados no deseados debido a las características de la lógica trivaluada que se define en el modelo relacional. Por ejemplo, con las filas en Profesor={<1,"Juan">} y AsignaturaImpartida={<1,NULL,0,1,2>}, se cumple el predicado anterior porque `creditos` es distinto de 2 (es NULL) y `responsable` no es igual a 1. Sin embargo, de acuerdo a la lógica trivaluada, la condición debería evaluarse a NULL (información desconocida).

Para soportar estas características, es necesario proporcionar definiciones adicionales para los operadores relacionales, de forma que tengan en cuenta la presencia de valores nulos. Estos predicados se definen en Alloy como una función que devuelve cierto, falso o NULL dependiendo de la presencia o no del valor NULL en los operandos en el momento de su evaluación. A continuación se muestra la especificación para el operador relacional de igualdad. Para el resto de operadores relacionales, la definición sería similar.

```
open util/boolean //Manejo de tipos True y False
fun eq [x:Int+NULL,y:Int+NULL]:(Bool + NULL){
  (isNotNULL[x] and isNotNULL[y]) =>
      (x=y => True else False) else NULL
}
```

Así, la regla C2 quedaría codificada como sigue:

```
some p:Profesor.pfilas,a:AsignaturaImpartida.afilas |
(eq[pid[p],pref[a]]=True) and
((eq[creditos[a],2]=False) and (eq[responsable[a],1]=False))
```

3.3 Generación de instancias de la base de datos

Una vez modelado el esquema de la base de datos (sección 3.1) y cada una de las reglas de cobertura generadas a partir de la consulta a probar (sección 3.2), se ejecutará el analizador de Alloy con el objetivo de generar una instancia de la base datos de forma que ésta satisfaga todas las reglas de cobertura. Esta instancia representa una carga de prueba para la base datos que satisface a su vez el criterio SQLFpc.

Para ello, todos los predicados que definen a cada una de las reglas de cobertura serán encapsulados en un único predicado en Alloy, de forma que en única ejecución del analizador se generen los datos en las tablas que cubran todas las reglas. La siguiente especificación ilustra lo anterior para un esquema formado por dos relaciones R y S y n reglas de cobertura derivadas de la consulta a probar.

```
pred test [t1:R,t2:S] {
  <Predicado correspondiente a la regla de cobertura 1>
…
  <Predicado correspondiente a la regla de cobertura n>
}
```

La ejecución del analizador con el comando `run test` generará filas en cada una de las tablas para cumplir todos los predicados.

Aunque la exposición se ha limitado a una única consulta a probar, el enfoque anterior es válido y aplicable a más de una consulta SQL, cada una de ellas con varias reglas de cobertura: un único predicado encapsula todas las restricciones que representan a todas las reglas de cobertura de todas las consultas. De esta forma, se instruye al analizador para generar un único conjunto de datos cubriendo diferentes situaciones de prueba para varias consultas SQL.

4 Caso de Estudio

Con el objetivo de mostrar el uso del enfoque desarrollado en este artículo, se va a generar una base de datos de prueba a partir de las consultas SQL extraídas de una aplicación real. La aplicación seleccionada, denominada HelpDesk, gestiona en la Universidad de Oviedo solicitudes de servicio y propuestas de cambios de software y hardware corporativo de usuarios finales, así como los distintos estados (anotaciones) por los que puede pasar una solicitud. Todos los experimentos se han realizado sobre un PC Intel® Core ™2 Duo a 2GHz. con 3GB. de memoria, utilizando la versión 4.0 de Alloy.

Para el caso de estudio se ha extraído una muestra de las consultas SQL de la aplicación con diferente complejidad y se ha considerando únicamente la parte del esquema de la base de datos que se utiliza en las consultas seleccionadas. En la Tabla 1 se muestran las tablas con las columnas que las componen, el tipo de cada una, las restricciones de valores NULL, las claves primarias (PK) y las claves ajenas (FK), junto con la columna a la que hacen referencia.

Tabla 1. Descripción de las tablas

Tabla	Columnas (nombre, tipo, restricciones de valores NULL , PK o FK)			
Usuario	ID	int,	NOT NULL,	PK
	IDArea	int,	NULL,	FK referencia a Usuario.ID
Solicitud	ID	int,	NOT NULL,	PK
	IDReceptor	int,	NOT NULL,	FK referencia a Usuario.ID
	IDPropietario	int,	NOT NULL,	FK referencia a Usuario.ID
Anotacion	ID	int,	NOT NULL,	PK
	IDSolicitud	int,	NOT NULL,	FK referencia a Solicitud.ID
	AntIDPropietario	int,	NULL,	FK referencia a Usuario.ID

Las consultas extraídas para el caso de estudio se muestran en la Tabla 2 (donde los parámetros se representan con la notación ?Num?) junto con el número de reglas de cobertura generadas (situaciones de prueba a cubrir de acuerdo al criterio SQLFpc) para cada una (20 en total).

Tabla 2. Código SQL de las consultas seleccionadas y número de reglas generadas

Id	Código SQL	NºReglas

Q1	`SELECT id FROM Solicitud`
	`WHERE (ID=?1?) AND`
	`((IDReceptor=?2?) OR (IDPropietario=?2?))`
		5

Q2	`SELECT s.id`
	`FROM Solicitud s LEFT JOIN Anotacion a ON`
	`s.ID=a.IDSolicitud`
	`WHERE (s.ID=?1?) AND ((s.IDReceptor=?2?) OR`
	`(s.IDPropietario=?2?) OR (a.AntIDPropietario=?2?))`
		8

Q3	`SELECT s.id`
	`FROM Solicitud s LEFT JOIN Anotacion a ON`
	`s.ID=a.IDSolicitud`
	`LEFT JOIN Usuario u ON a.AntIDPropietario=u.ID`
	`WHERE (s.ID=?2?) AND (u.IDArea=?1?)`
		7

Siguiendo el procedimiento descrito en la sección 3, se ha generado la especificación Alloy para el esquema de datos de la Tabla 1 y las restricciones para las reglas de cobertura de las consultas de la Tabla 2 encapsuladas en un único predicado.

La ejecución del analizador con la especificación anterior genera una instancia de la base de datos (Tabla 3) que permite satisfacer el conjunto de las reglas de cobertura para todas las consultas. La base de datos contiene 3 usuarios, 5 solicitudes y 5 anotaciones, y el tiempo de generación es de 7,6 segundos.

Para validar los datos de prueba generados, se ha cargado la instancia de base de datos de prueba y, utilizando la herramienta SQLFpc (https://in2test.lsi.uniovi.es/sqltools/sqlrules/), se ha evaluado la cobertura de las consultas. El porcentaje de cobertura para todas las consultas es del 100%. Este resultado es el esperado, ya que en el propio proceso de generación se había indicado que todas las restricciones se debían cumplir.

Tabla 3. Instancia de la base de datos generada automáticamente

	Usuario <ID,IDArea>	**Solicitud** <ID,IDReceptor,IDPropietario>	**Anotacion** <ID,IDSolicitud,AnIdPropietario>
Filas	<4, 5>	<1, 5, 5>	<3, 7, 7>
	<5, 4>	<3, 4, 4>	<4, 7, 4>
	<7, NULL>	<4, 5, 4>	<5, 5, 4>
		<5, 5, 5>	<6, 1, 7>
		<7, 4, 7>	<7, 3, NULL>

La base de datos generada se ha comparado con una parte de la base de datos del HelpDesk (que contiene 501 solicitudes, 1.304 anotaciones y 261 usuarios, y a la que se hará referencia como base de datos de producción) desde dos puntos de vista: la cobertura alcanzada de acuerdo al criterio SQLFpc y la efectividad de los casos de prueba utilizando una técnica de generación de mutantes.

Los resultados de la comparación se muestran en la Tabla 4. La columna %SQLFpc indica, para cada consulta, el número de reglas de cobertura generadas y los porcentajes de cobertura obtenidos con la base de datos de producción y la generada automáticamente. Como se observa, con la base de datos de producción se alcanza un 85,00% de cobertura frente al 100% de cobertura de la base de datos generada. La columna %SQLMut indica, para cada consulta, el número de mutantes no equivalentes generados y el porcentaje de mutantes muertos con la base de datos de producción y la generada, respectivamente. Para la generación de los mutantes se ha utilizado la herramienta SQLMutation (https://in2test.lsi.uniovi.es/sqlmutation/) [13]. El porcentaje de mutantes muertos es de 87,00% y 82,35% con la base de datos de producción y la generada, respectivamente.

Como se puede observar, la disminución del porcentaje de mutantes muertos con la base de datos generada respecto a la de producción es muy pequeña teniendo en cuenta el reducido número de filas de la base de datos generada automáticamente.

Así mismo, comparando la base de datos de producción con la base de datos generada cabe destacar que en el segundo caso el número de filas en cada tabla es

muy inferior (pasando de 501 a 5 solicitudes y de 1.304 a 5 anotaciones), alcanzando además el 100% de cobertura.

Tabla 4. Porcentajes de cobertura SQLFpc y de mutantes muertos utilizando la base de datos de producción y la generada de forma automática.

		%SQLFpc			%SQLMut	
Id	**N. Reglas**	**Producción**	**Generada**	**N. Muts**	**Producción**	**Generada**
Q1	5	100%	100%	71	88,73%	84,51%
Q2	8	87,50%	100%	139	89,93%	89,21%
Q3	7	71,43%	100%	113	82,30%	72,57%
Total	**20**	**85,00%**	**100%**	**323**	**87,00%**	**82,35%**

5 Conclusiones

Se ha presentado un enfoque que permite la generación automática de bases de datos de prueba para consultas SQL. La aplicación a un caso de estudio real con varias consultas SQL muestra buenos resultados preliminares, ya que es posible generar una única carga de pruebas de tamaño reducido para todas las consultas que cumple un conjunto importante restricciones. Dado un conjunto de consultas SQL sobre un esquema de base datos, el esquema y las diferentes situaciones de prueba (reglas de cobertura) sobre las consultas se representan como un conjunto de restricciones, que se resuelven posteriormente con el *constraint solver* Alloy.

Las limitaciones del enfoque propuesto vienen impuestas principalmente por el analizador Alloy. Debido a que Alloy enumera todas las posibles instancias de la base de datos, no es posible manejar grandes bases de datos y por ello se representa únicamente la parte del esquema de la base de datos que interviene en cada consulta. Además, los tipos de datos manejados se restringen a enteros, por lo que será necesario utilizar valores simbólicos para otro tipo de datos diferente.

Los resultados de este trabajo pueden ser ampliados en varias direcciones. Por una parte, extendiendo el tipo de consultas SQL para dar soporte a otras características habituales en SQL, como por ejemplo agrupamientos y funciones de agregación. Otra es relativa al soporte automático, de forma que el proceso completo de generación de pruebas (obtención de reglas de cobertura, generación de datos de prueba y carga posterior de la base de datos) quede integrado en una única herramienta.

Agradecimientos. Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación y Fondos FEDER (Proyecto TIN2007-67843-C06-01), el Gobierno del Principado de Asturias (Proyecto CN-07-168) y el Gobierno de Castilla-La Mancha (Proyecto PCI-08-121-1374).

Referencias

1. Binnig, C., Kossmann, D., Lo, E.M., Oztsu, T.: Qagen: generating query-aware test databases. In SIGMOD Conference, pp 341--352. (2007)
2. Bruno,N., Chaudhuri,S.: Flexible database generators. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), pp. 1097--107. (2005).
3. Chilenski, J.J.: An investigation of three forms of the modified condition decision coverage (MCDC) criterion. Technical Report DOT/FAA/AR-01/18, U.S. Department of Transportation, Federal Aviation Administration. (2001)
4. Emmi, M., Majumdar, R.,Sen, K.: Dynamic Test Input Generation of Database Applications. In: Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), pp. 151--16. (2007).
5. IBM DB2 Generator. http://www-01.ibm.com/software/data/db2imstools/products/db2-luw-tools.html.
6. Jackson, D.: Alloy: A lightweight object modelling notation. ACM Transactions on Software Engineering and Methodology, 11 (2), 266—290. (2002)
7. Jackson, D., Schechter, I., Shlyakhter, I.: Alcoa: the alloy constraint analyzer (2000). In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 730—733. (200)
8. Kapfhammer, G.M., Soffa, M.L.: A Family of Test Adequacy Criteria for Database-Driven Applications. In: Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp 98—107. (2003)
9. Khalek, S.A., Elkarablieh, B., Laleye, Y.O., Khurshid, S.: Query-aware Test Generation Using a Relational Constraint Solver. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 238—247. (2008)
10. Smaragdakis, Y., Csallner, C., Subramanian, R.: Scalable automatic test data generation from modeling diagrams. In: Proceedings of EEE/ACM international Conference on Automated Software Engineering (ASE), pp. 4—13. (2007)
11. Suárez-Cabal, M.J., Tuya J.: Structural Coverage Criteria for Testing SQL Queries. Journal of Universal Computer Science, 15 (3), 584—619. (2009)
12. Tuya, J., Suárez-Cabal, M.J., de la Riva, C.: A practical guide to SQL white-box testing. ACM SIGPLAN Notices; 41(1). (2006)
13. Tuya, J., Suárez-Cabal, M.J., de la Riva, C.: Mutating Database Queries. Information and Software Technology, 49 (4): 398-417. (2007)
14. Tuya, J., Suárez-Cabal, M.J., de la Riva, C.: Full Predicate Coverage for Testing SQL Database Queries. Software Testing, Verification and Reliability (under revision).(2009)
15. Willmor, D., Embury, S.M.: An intensional approach to the specification of test cases for database applications. In: Proceeding of the International Conference on Software Engineering (ICSE), pp.102—111. (2006)
16. Zhang, J., Xu, C., Cheung, S.C.: Automatic Generation of Database Instances for White-box Testing. In: Proceedings of the International Computer Software and Applications Conference (COMPSAC). (2001)

Selección de Características para Mejorar los Modelos de Verificación de Información en EAI *

I. Fernández de Viana[1], J.L. Arjona[1], J.L. Álvarez[1], and P. Abad[1]

Dpto. Tecnologías de la Información
E.P.S. La Rábida
Universidad de Huelva
21071 La Rábida (Huelva)
{i.fviana, jose.arjona, alvarez, abadhe}@dti.uhu.es

Resumen Abaratar los costes de mantenimiento de soluciones de Integración de Aplicaciones Empresariales (EAI) se convierte en un reto cuando lo que intentamos integrar son aplicaciones web amigables. La solución a este problema consiste en el uso de sistemas automáticos que permitan navegar hasta la información de interés, extraerla, estructurarla y verificarla. La verificación permite caracterizar la información extraída en base a modelos complejos que son usados para comprobar si la información es válida. En este artículo demostramos empíricamente cómo la selección de características permite simplificar y mejorar los modelos que se obtienen para la verificación de información.

1. Introducción

La Web es la principal fuente de información y ha sido concebida para ser consumida por seres humanos, lo que supone una desventaja si lo que pretendemos es su procesado automático. Y es que no es habitual que los sitios web proporcionan una interfaz programática para acceder a una vista estructurada de la información de interés que nos ofrecen.

En el contexto de EAI, la integración de la información contenida en un sitio web con otras aplicaciones empresariales, pasa por disponer de un conjunto de APIs que proporcionen la posibilidad de interactuar con el sitio como si de un ser humano se tratara. Estas APIs deberían de aceptar una serie de consultas definidas en un lenguaje estructurado de alto nivel, como SQL o SPARQL[1], rellenar formularios a partir de ellas [1] [2], ejecutarlos y navegar a través de los resultados [3], devolver las páginas que contengan la información de interés, extraer dicha información [4] y dotarla de estructura de acuerdo con la ontología que estemos utilizando [5] y, por último, comprobar que la información obtenida es correcta [6] [7].

Al elemento que se encarga de realizar esta última actividad se le denomina **verificador** y es de vital importancia ya que, de no existir, podríamos alimentar a las distintas aplicaciones con datos no veraces. Esto tendría unos resultados imprevisibles sobre los procesos (como, por ejemplo, de tomas de decisión) que con posterioridad los usen.

* Este trabajo ha sido parcialmente financiado por el proyecto INTEGRAWEB: TIN2007-64119 (MICINN), P07-TIC-2602 y P08-TIC-4100 (CICE, Junta Andalucia).

[1] http://www.w3.org/TR/rdf-sparql-query/

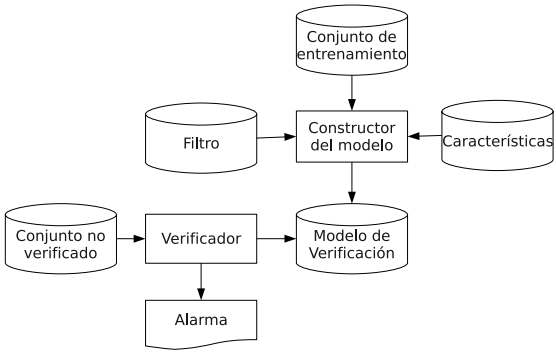


Figura 1. Marco general del proceso de verificación

La figura 1 ilustra el marco general del proceso de verificación. A partir de los datos contenidos en nuestro **conjunto de entrenamiento** creamos un modelo matemático que sirve para estudiar el comportamiento global de las entidades que forman dicho conjunto. Para crearlo cuantificamos algunas de sus cualidades gracias a una serie de características. Para que este proceso sea lo más veraz posible, aplicamos **filtros** que eliminen valores extraños, *outliers*, del conjunto y elegimos las características que mejor caractericen al conjunto. Una vez construido el **modelo de verificación**, el verificador se encargará de comprobar si el **conjunto de datos a verificar** tiene las mismas cualidades que el modelo de verificación. Si no las presenta entonces el verificador tendrá que lanzar una alarma que conllevará la intervención de un experto.

Los principales modelos de verificación son los propuestos en [8] [6] [7] [9], todos se basan en el uso de diversas herramientas estadísticas y probabilísticas que les permiten crear los modelos y combinar los datos que estos tienen. En este artículo presentamos cómo mejorar la calidad de los resultados logrados por la propuesta de verificación de Kushmerick desde el punto de vista de la selección de características. Esto es, mejoraremos el modelo de verificación haciendo un estudio previo que permita vislumbrar qué conjunto de características caracterizan mejor al conjunto de entrenamiento.

Este trabajo se ha organizado en las siguientes secciones. En la Sección 2 se presentarán los diversos modelos de verificación existentes haciendo especial énfasis en clasificar los distintos tipos de características propuestas. En la Sección 3 introduciremos en qué consisten la selección de características y resumiremos algunas técnicas. En la Sección 4 explicaremos la metodología de experimentación seguida. En la Sección 5 expondremos los resultados obtenidos. Finalmente, reservaremos la Sección 6 para conclusiones y trabajos futuros.

2. Verificación de información

Para poder comprobar la veracidad de un conjunto de datos necesitamos dos elementos fundamentales: las características y los modelos de verificación.

2.1. Características

Las características son atributos, cualidades o propiedades cuantificables de los distintos elementos que forman nuestros conjuntos de datos y se clasifican según su ámbito de aplicación y su dominio. Según su ámbito distinguimos entre características de slot o de conjunto de datos. Para nosotros un slot será un atributo o un registro presente en el conjunto de datos (ver figura 2). Por ejemplo, el número de registros que forman un conjunto de datos es una característica de dicho conjunto mientras que la longitud de las palabras extraídas sería de atributo.

Según su dominio tendremos características numéricas y características categóricas.

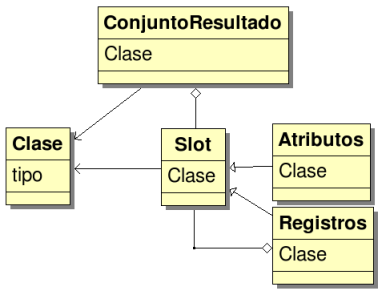


Figura 2. Slots, registros y atributos

Características Numéricas. Las características numéricas transforman los valores de los slots o los conjuntos de datos en números. Estas características se suelen agrupar en cinco grupos denominados:

- **Conteo:** que van desde aquellas que cuentan el número de slots de una clase que hay en un conjunto de datos, hasta las que cuentan el número de slots que cumplen una serie de restricciones o que están caracterizados por ciertos patrones a izquierda y derecha [8].
- **Léxicas:** se centran en características de caracteres y suelen estar relacionadas con la codificación usada para representarlos. En [10] se recogen hasta 42 características del tipo longitud de palabras, conteo de palabras, media o longitud de palabras, etc.
- **Similitud:** comprueban el grado de igualdad entre atributos y el conjunto de atributos de una clase determinada [11]. Algunos autores intentan medir la similitud entre registros desde un punto de vista estructural. Como los registros se pueden ver como árboles ordenados y etiquetados, en [12] se usan medidas de edición de árboles para calcularla.
- **Diseño:** analizan las posiciones que ocupan los slots que forman el mismo registro o adyacentes en el texto HTML. En [9] hay diversas propuestas que se engloban dentro de este tipo de clase de características.

- **Tendencia:** miden las fluctuaciones en los valores de las características numéricas de un registro [9].

Características Categóricas. Las características categóricas van desde patrones que describen la estructura de registro, hasta restricciones matemáticas sobre los valores de algunos atributos. Estas características se pueden clasificar en:

- **Estructura del registro:** analizan los registros desde un punto de vista estructural. Un registro está formado por un conjunto fijo en número y tipo de atributos, estas técnicas se encargan de comprobar que este tipo de restricciones se cumpla [9].
- **Clases lingüísticas:** estudian las características lingüísticas de un atributo. En [10] esto se traduce en clasificar los atributos como sustantivos, verbos, nombre, apellidos, etc.
- **Expresiones regulares:** caracterizan todo el contenido de los atributos de una clase mediante patrones [10], o sólo se tienen en cuenta los patrones de comienzo y de fin [8].
- **Restricciones semánticas:** buscan restricciones entre los atributos de diversas clases relacionadas. En [13] se presenta un motor de inferencia capaz de inferir este tipo de restricciones para un conjunto de datos.

2.2. Modelos

Los valores que toman las características aplicadas tanto a los slots como a los conjuntos de datos son usados por el verificador como evidencia para determinar si un conjunto de datos es válido o no. Estos valores no se consideran individualmente sino en su conjunto.

En los siguientes apartados veremos las principales propuestas existentes tanto para combinar dichos valores como para determinar en qué grado este valor nos servirá para establecer la validez o no de los datos, esto es, si el verificador tiene que lanzar una alarma.

Modelo de Kushmerick. Este autor presenta dos modelos que son prácticamente idénticos [6,7]. El conjunto de entrenamiento que usa para crear el modelo de verificación sólo tiene conjuntos de datos válidos. En su propuesta inicial trabajaba con las siguientes características: número de registros por conjunto de datos, número de palabras, longitud, fracción de dígitos, letras, letras mayúsculas, letras minúsculas, símbolos de puntuación y símbolos HTML. Posteriormente también incluyó características relacionadas con la estructura de los registros y el número de registros de cada una de las clases que hay en cada registro.

Estas características son modeladas como si de variables aleatorias que siguen una distribución de probabilidad se trataran. Para las variables numéricas sugiere usar una distribución normal mientras que para las categóricas usa una empírica que crea basándose en los datos que estas variables toman en el conjunto de entrenamiento.

A la hora de verificar un conjunto de datos, primero calcula los valores que toman las distintas características en dicho conjunto. Luego obtiene la probabilidad con la

que estos valores siguen las distribuciones antes calculadas. Para combinar estas probabilidades se sugieren tres alternativas denominadas de independencia, vinculación y equivalencia.

El valor que se obtiene de aplicar cualquiera de estos métodos es considerado una muestra de una distribución de probabilidad V. Dicha distribución se calculó tomando como muestras las probabilidades con las que los datos del conjunto de entrenamiento siguen las distribuciones normales. Solo se lanza la alarma si la probabilidad con la que este valor sigue la distribución V es inferior a un límite.

Modelo de Lerman. Este modelo [8] parte de un conjunto de entrenamiento formado únicamente por datos válidos. Junto a las características utilizadas por Kushmerick, Lerman usa un algoritmo propio, denominado DATAPROG, para inferir los patrones de comienzo y fin y calcula el número de atributos de una clase que cumplen alguno de los patrones aprendidos para la clase.

Tanto el conjunto de entrenamiento, como cada uno de los conjuntos que se deseen verificar, están representados por un vector que contiene los valores medios de cada una de las características.

Para verificar un conjunto de datos no verificado comprueba si su vector y el del conjunto de entrenamiento son estadísticamente iguales usando el estadístico de Pearson. Si el valor devuelto está por debajo de un límite entonces se dispara una alarma.

Modelo de MacCann. Este modelo de verificación [9] considera las siguientes características a nivel de conjunto de datos: número de atributos de cada clase, longitud media, número medio de palabras, longitud media de las palabras, densidad de dígitos, densidad de palabras alfabéticas, numéricas y alfanuméricas, orden de tuplas atributo y una serie de características booleanas que indican el cumplimiento de restricciones semánticas definidas por el usuario. Además, para cada una de estas características, calcula su tendencia con el objetivo de detectar fluctuaciones en sus valores.

El proceso de verificación propuesto se basa en cuatro puntos fundamentales:

1. Las estimaciones de las características se calculan mediante densidades normalizas. Esto es, calcula la probabilidad de que una característica tenga un valor con más probabilidad que otro.
2. Usa tanto conjuntos de valores válidos como inválidos. Cada característica se perfila mediante dos distribuciones normalizadas, una para los datos válidos y otra para los inválidos. Aunque los autores proponen dos formas de combinar dichas distribuciones, los resultados empíricos que ellos presentan en su artículo demuestran que ninguna de ellas presenta unos resultados mucho mejores que el resto.
3. Calcula el valor combinado de las distintas características teniendo en cuenta el peso discriminatorio de cada una de ellas tanto para los conjuntos de datos válidos como los inválidos.
4. El usuario define un límite fijo que se usará para lanzar la alarma.

3. Técnicas de selección de características

Los algoritmos de aprendizaje basados en instancias son muy susceptibles a características irrelevantes ya que toman sus decisiones basándose en los datos contenidos en el conjunto de entrenamiento. Por tanto, antes de aplicar alguno de estos algoritmos es necesario aplicar algún tipo de preprocesamiento, en general, y de selección de características [14] en particular.

La selección de características se puede definir como un proceso que elige un conjunto mínimo de M características de entre un conjunto original de N ($M \leq N$). Intenta que el espacio de características sea óptimamente reducido de acuerdo con un criterio de evaluación. Como en general N es muy grande, encontrar el mejor subconjunto de características es un problema **NP**-completo.

El proceso de selección de características consiste en cuatro pasos básicos (figura 3):

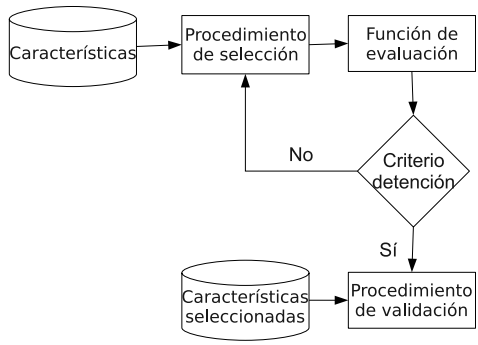


Figura 3. Selección de características

- **Procedimiento de selección**. Se selecciona el posible subconjunto de características. Si se dispone de subconjuntos de N características este proceso debería generar, en el peor de los casos, 2^N conjuntos diferentes evaluables.
- **Función de evaluación**. Se evalúa el subconjunto de características.
- **Criterio de detención**. Se comprueba si el subconjunto cumple diversas características deseables.
- **Procedimiento de validación**. Se valida la calidad del mejor subconjunto de características seleccionado, es necesario conocer el dominio de aplicación.

Existen gran cantidad de técnicas de selección de características como puede ser BFF, MDLM, Focus, FSBC, Relief, CFS, Set Cover, Mitra's, ELSA, LVI, etc. Podemos consultar [15,16] para un estudio más detallado de cada una de estas técnicas.

3.1. Selección de subconjuntos

Si atendemos al procedimiento de selección de subconjuntos, la naturaleza de este proceso viene determinado por dos elementos: el punto de comienzo de la búsqueda y la estrategia de búsqueda.

La búsqueda puede comenzar con el conjunto vacío e ir añadiendo características (**búsqueda hacia delante**), o empezar con el conjunto completo de características e ir quitando características (**búsqueda hacia atrás**). Incluso podemos partir de ambos conjuntos (o un conjunto aleatorio de características) a los que iremos quitando o añadiendo características (**búsqueda bidireccional**).

Las diversas estrategias de búsqueda nos permiten explotar el espacio de búsqueda de diferentes maneras. Si usamos una **búsqueda completa**, como *branch and bound* [17], tenemos garantizado un óptimo global respecto al criterio de evaluación usado. También podemos optar por una **búsqueda secuencial**, como son las aproximaciones basadas en técnicas *greedy* [18], reducimos el área de búsqueda y, por tanto, aumentamos el riesgo de caer en máximos locales. Por último, tenemos la **búsqueda aleatoria**, como *simulated annealing* [19], que va generando nuevos conjuntos de características siguiendo un patrón aleatorio y así intentan reducir el espacio de búsqueda y evitar caer en máximos locales.

3.2. Evaluación de subconjuntos

Si ahora nos fijamos en la función de evaluación, hablaremos de tres tipos de técnicas de selección de características. Los **métodos de filtrado o independientes** evalúan la bondad de una característica o conjunto de ellas basándose en las características intrínsecas del conjunto de entrenamiento. Los diversos algoritmos que se usan están basados en medidas de distancia, información, dependencia y/o consistencia.

Los **métodos de envoltura o dependientes** hacen uso de un algoritmo de minería de datos para realizar la evaluación. Se obtienen mejores resultados que en el caso anterior, pero traen consigo un costo computacional mucho mayor.

Por último, los **métodos mixtos** intentan aprovechar las ventajas de los dos modelos anteriores. Durante diferentes fases del proceso de búsqueda, explotan la forma totalmente diferente de evaluar los subconjuntos de características de los dos métodos anteriores.

Respecto a la evaluación hay que tener en cuenta que la salida generada por los distintos algoritmos varía. Esto es, algunos devuelven un conjunto mínimo de características y otros devuelven la lista completa de características ordenadas. Si disponemos de un subconjunto mínimo de características es difícil eliminar alguna de ellas ya que no disponemos de información sobre su capacidad discriminatoria individual. Este problema no aparece si el algoritmo nos devuelve un ranking de características en donde podemos eliminar las más irrelevantes, aunque sí es cierto que hay que establecer un umbral de corte.

3.3. Criterios de detención

Otro aspecto importante es establecer el criterio de parada de nuestro algoritmo de selección de características. Existen criterios tan variopintos como una búsqueda comple-

ta, establecer unos límites como un número mínimo de características o de iteraciones, finalizar la búsqueda si sucesivos subconjuntos de características no mejoran lo que ya teníamos, etc.

3.4. Procedimiento de validación

Una vez seleccionado el mejor conjunto de características es necesario comprobar si dicho subconjunto permite resolver el problema de forma óptima. Hay que recordar que el criterio de evaluación (salvo para el caso de los algoritmos de envoltura) no tiene por qué ser el mismo al usado por la técnica concreta de resolución que usemos.

4. Metodología de experimentación

Como nuestra experimentación se centra en estudiar la influencia de la selección de características en el modelo de Kushmerick, usaremos los mismo conjuntos de datos que aparecen en [6,7]. De entre los 27 sitios que cita el autor usaremos 14 de ellos, que cubren los dominios más representativos, para realizar el estudio.

Para elegir los algoritmos de selección que mejor se adapten al problema aplicamos la metodología expuesta en [20]. De un estudio previo de los datos se aprecia que son de tipo numérico, que hay muchas más instancias que características, que todas las características son aplicables a todos los datos y no están sujetos a problemas de ruido. Como también disponemos de información de la clase, sólo debemos considerar técnicas usadas en problemas de clasificación y no de agrupamientos y, por lo tanto, algoritmos como SBUD, Mitra's o ELSA quedan descartados.

Nos centraremos únicamente en los algoritmos de filtro pero dentro de ellos hemos usado técnicas que calculan la bondad de los subconjuntos de variables atendiendo a medidas de distancias. información, dependencia y consistencia.

Respecto al tipo de búsqueda, como el número de características a aplicar es muy superior al que esperamos seleccionar deberíamos optar por una búsqueda hacia delante. Además, como no tenemos restricciones en cuanto al tiempo de cómputo optaremos por hacer una búsqueda completa. El cuadro 1 resume los algoritmos de selección de características usados.

Método	Tipo	Evaluador	Medida	Búsqueda
1	Filtro	Informaction Gain Atribute Ranking [21]	Información	Completa
2	Filtro	Relief [22]	Distancia	Completa
3	Filtro	CFS [23]	Dependencia	Completa
4	Filtro	Consistency-based Subset Evaluator [24]	Consistencia	Completa

Cuadro 1. Algoritmos de selección evaluados

El algoritmo *Informaction Gain Atribute Ranking* no devuelve un subconjunto mínimo de características sino un ranking de las mismas. En cambio, cuando se vaya a

construir el modelo de verificación es preciso conocer las características que usaremos para crearlo y no su ranking. Para este caso procederemos de la misma manera que se indica en [15].

Una vez que tenemos los subconjuntos de características es necesario validarlos (figura 3). Para hacer esto usaremos el modelo de Kushmerick y repetiremos exactamente la misma experimentación descrita en [7] pero ahora no crearemos los modelos con las 9 características que él propone sino que aplicaremos las técnicas de selección de características mencionadas para que nos den el mejor subconjunto de características para cada uno de los conjuntos de entrenamiento.

En la experimentación usamos WEKA (Waikato Environment for Knowledge Analysis) para aplicar los algoritmos de selección de características y la implementación proporcionada por el propio Kushmerick del algoritmo RAPTURE.

5. Resultados

El cuadro 2 muestra el número de clases contenidas en cada uno de los sitios web y el número de características que ha seleccionado cada algoritmo utilizado en la experimentación. Hemos omitido el último método de selección ya que para ninguno de los casos hemos logrado que convergiera dentro de los recursos que considerábamos óptimos.

Puede observarse como la reducción del número de características es muy significativa en la mayoría de los casos, obteniéndose una reducción media del 34 %. En particular las características seleccionadas dependen de cada sitio y, en general, del dominio de aplicación sin que exista un subconjunto de características común para todo el conjunto de entrenamiento.

Instancia	Clases	Método 1	Método 2	Método 3
ALTA	3	3	6	3
BIBL	3	4	3	2
CDPL	8	2	5	4
CESP	4	2	9	2
CINE	2	2	4	1
COLC	2	2	2	1
CORL	7	2	2	4
EXPE	4	2	2	2
FOR5	2	2	2	1
INAF	3	2	2	2
IRTI	4	2	3	2
LYCO	4	4	7	4
META	4	3	6	5
MONJ	5	2	5	5

Cuadro 2. Clases y características seleccionadas

Tras realizar la reducción, el cuadro 3 resumen los resultados obtenidos al validar con el algoritmo RAPTURE, para cada uno de los sitios web. Las medidas mostradas se corresponden con los valores de corrección (A), cobertura (R), precisión (P) y medida F (F). La comparativa se establece entre los conjuntos de datos originales, conteniendo las 9 características, y cada uno de los conjuntos reducidos por cada uno de los algoritmos de selección.

Instancia	Original				Método 1				Método 2				Método 3			
	A	P	R	F	A	P	R	F	A	P	R	F	A	P	R	F
ALTA	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
BIBL	0,8	1	0,8	0,89	0,8	1	0,8	0,89	0,8	1	0,8	0,89	0,8	1	0,8	0,89
CDPL	1	1	1	1	1	1	1	1	0,8	1	0,8	0,89	1	1	1	1
CESP	0,86	1	0,86	0,92	0,86	1	0,78	1	0,86	1	0,86	0,92	0,85	1	0,85	0,92
CINE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
COLC	0,61	1	0,61	0,76	1	1	1	1	1	1	1	1	1	1	1	1
CORL	0,83	1	0,83	0,9	1	1	1	1	1	1	1	1	1	1	1	1
EXPE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FOR5	0,8	1	0,8	0,89	0,8	1	0,8	1	0,8	1	0,8	0,89	0,8	1	0,8	0,89
INAF	0,79	1	0,79	0,88	0,71	1	0,65	1	1	1	1	1	1	1	1	1
IRTI	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LYCO	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
META	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MONJ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Cuadro 3. Resultados

En general, la selección mejora los resultados obtenidos por el conjunto de datos original en el 19 % de los casos, y es capaz de igualarlos en el 71.4 % de los mismos. Resulta significativo mostrar que la mejora obtenida, cuando esta se produce, está por encima del 24 %. Por el contrario, los resultados obtenidos son peores, de media, en el 9,5 % de los casos.

De forma individual, los algoritmos 2 y 3, obtienen una proporción de 21,4 % de mejora, 71,4 % de igualdad y 7,2 % de empeoramiento con respecto al conjunto de datos original. Por su parte, el algoritmo 1 arroja unos resultados de 71,4 % de igualdad y 14,3 % de mejora y empeoramiento con respecto a los datos originales.

Es importante destacar que, para todos los sitios web, siempre alguno de los algoritmos de selección de características consigue, al menos, igualar los resultados obtenidos con el conjunto de datos original. Así pues, para cada sitio web, alguno de los tres algoritmos estudiados ofrece mejores o iguales resultados.

6. Conclusiones y trabajos futuros

En este artículo hemos estudiado cómo la aplicación de técnicas de selección de características incide directamente en la calidad de los resultados obtenidos por los modelos

de verificación. El estudio empírico ha demostrado que esta afirmación es cierta ya que hemos obtenido modelos de verificación mejores y más sencillos tomado como referencia el modelo propuesto por Kushmerick.

En futuros trabajos ampliaremos el estudio a otros modelos de verificación y usaremos métodos de selección de características híbridos que permitan obtener mejores subconjuntos de características.

Referencias

1. Raghavan, S., Garcia-Molina, H.: Crawling the hidden web. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2001) 129–138
2. Pan, A., Raposo, J., Álvarez, M., Carneiro, V., Bellas, F.: Automatically maintaining navigation sequences for querying semi-structured web sources. Data Knowl. Eng. **63**(3) (2007) 795–810
3. Lage, J.P., da Silva, A.S., Golgher, P.B., Laender, A.H.F.: Automatic generation of agents for collecting hidden web pages for data extraction. Data Knowl. Eng. **49**(2) (2004) 177–196
4. Kayed, M., Shaalan, K.F.: A survey of web information extraction systems. IEEE Trans. on Knowl. and Data Eng. **18**(10) (2006) 1411–1428 Member-Chia-Hui Chang and Member-Moheb Ramzy Girgis.
5. Arjona, J.L., Corchuelo, R., Ruiz, D., Toro, M.: From wrapping to knowledge. IEEE Trans. Knowl. Data Eng. **19**(2) (2007) 310–323
6. Kushmerick, N.: Regression testing for wrapper maintenance. In: AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence (1999) 74–79
7. Kushmerick, N.: Wrapper verification. World Wide Web **3**(2) (2000) 79–94
8. Lerman, K., Minton, S.N., Knoblock, C.A.: Wrapper maintenance: A machine learning approach. Journal of Artificial Intelligence Research **18** (2003) 2003
9. McCann, R., AlShebli, B., Le, Q., Nguyen, H., Vu, L., Doan, A.: Mapping maintenance for data integration systems. In: VLDB '05: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment (2005) 1018–1029
10. Chidlovskii, B., Roustant, B., Brette, M.: Documentum eci self-repairing wrappers: performance analysis. In: SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2006) 708–717
11. Wong, T.L., Lam, W.: Adapting web information extraction knowledge via mining site-invariant and site-dependent features. ACM Trans. Interet Technol. **7**(1) (2007) 6
12. Bille, P.: A survey on tree edit distance and related problems. Theor. Comput. Sci. **337**(1-3) (2005) 217–239
13. Ernst, M.D., Perkins, J.H., Guo, P.J., McCamant, S., Pacheco, C., Tschantz, M.S., Xiao, C.: The Daikon system for dynamic detection of likely invariants. Science of Computer Programming **69**(1–3) (December 2007) 35–45
14. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analysis **1** (1997) 131–156
15. Hall, M.A., Holmes, G.: Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data Engineering **15**(6) (2003) 1437–1447
16. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3** (2003) 1157–1182

17. Narendra, P.M., Fukunaga, K.: A branch and bound algorithm for feature subset selection. IEEE Trans. Comput. **26**(9) (1977) 917–922
18. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Norwell, MA, USA (1998)
19. J. Doak: An Evaluation of Feature Selection Methods and Their Application to Computer Security. Technical report, Univ. of California at Davis, Dept. Computer Science (1992)
20. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. Knowledge and Data Engineering, IEEE Transactions on **17**(4) (2005) 491–502
21. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In: CIKM '98: Proceedings of the seventh international conference on Information and knowledge management, New York, NY, USA, ACM (1998) 148–155
22. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: ML92: Proceedings of the ninth international workshop on Machine learning, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1992) 249–256
23. Hall, M.A., Smith, L.A.: Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper (1999)
24. Almuallim, H., Dietterich, T.G.: Learning with many irrelevant features. In: In Proceedings of the Ninth National Conference on Artificial Intelligence, AAAI Press (1991) 547–552

Análisis de los Efectos de las Políticas de Gestión de la Capacidad de los Servicios en el Cumplimiento de los SLAs utilizando Simulación

Elena Orta[1], Mercedes Ruiz[1] y Miguel Toro[2]
[1] Departamento de Lenguajes y Sistemas Informáticos
Escuela Superior de Ingeniería
C/ Chile, 1
11003 – Cádiz, España
[2]Departamento de Lenguajes y Sistemas Informáticos
Escuela Técnica Superior de Ingeniería Informática
Avda. Reina Mercedes, s/n
41012 – Sevilla, España
{elena.orta, mercedes.ruiz}@uca.es, migueltoro@us.es

Resumen. Actualmente, cada vez son más las empresas proveedoras de servicios TI que recurren a ITIL (*Information Technology Infraestructure Library*) para gestionar sus servicios con la finalidad de mejorar la calidad y reducir el coste de los mismos. En este trabajo se propone un modelo dinámico de simulación aplicado en el ámbito del proceso *Gestión de la Capacidad* de ITIL que ayuda a decidir qué política de gestión de la capacidad de los servicios que los proveedores asignan a sus clientes es más adecuada respecto al cumplimiento de los SLAs (*Service Level Agreements*). El modelo permite estudiar los efectos que estas políticas tienen sobre el rendimiento de los servicios y sobre los costes que tendría que asumir el proveedor si los tiempos de respuesta no son los acordados en los SLAs. Con esta finalidad, se han realizado diferentes análisis de sensibilidad variando la capacidad del servicio, la forma de gestionarla y el porcentaje de la capacidad que se utiliza para responder las invocaciones al servicio en los tiempos de respuesta acordados.

Palabras clave: Gestión de servicios TI, Gestión de la capacidad, Modelado y Simulación

1 Introducción

El éxito de las empresas proveedoras de servicios TI depende cada vez más de que los servicios que ofrecen se correspondan con los objetivos de su negocio y satisfagan las expectativas de sus clientes. Por tanto, estas empresas no sólo deben centrarse en la tecnología y en su organización interna, sino que también deben tener en cuenta la calidad de los servicios y la relación que mantienen con sus clientes. En este contexto, una gestión eficaz de los servicios ayudará a alinearlos con las necesidades actuales y futuras de la propia organización y de sus clientes, a mejorar calidad y a reducir el coste de los servicios.

Actualmente, cada vez son más las empresas proveedoras de servicios TI que recurren a la Biblioteca de Infraestructura de Tecnología de la Información (*ITIL*, *Information Technology Infraestructure Library*) [12] para gestionar sus servicios. ITIL ofrece un marco de referencia de buenas prácticas en la gestión de servicios TI cuyo objetivo es proporcionar servicios de calidad.

ITIL V3 propone utilizar las técnicas de modelado y simulación en algunos de sus procesos como, por ejemplo, en los procesos de los módulos *Gestión Estratégica del Servicio* y *Diseño del Servicio*. En este trabajo se propone un modelo dinámico de simulación aplicado en el ámbito del proceso *Gestión de la Capacidad* del módulo *Diseño del Servicio* de ITIL. En concreto, se aplica a la problemática de decidir qué política de gestión de la capacidad de los servicios que los proveedores asignan a sus clientes es más adecuada respecto al cumplimiento de los SLAs. El propósito del modelo es estudiar los efectos que diferentes políticas de gestión tienen sobre el rendimiento de los servicios y sobre los costes que tendrían que asumir los proveedores si los tiempos de respuesta no son los acordados.

El trabajo se estructura de la siguiente manera: el apartado 2 ofrece una visión global del proceso *Gestión de la Capacidad* de ITIL. En el apartado 3 se presentan las principales técnicas de gestión de servicios TI y se referencian algunos trabajos actuales que estudian esta problemática. En el apartado 4 se describe el caso de estudio concreto en el que se aplica el modelo dinámico de simulación que se propone y se explican las características o aspectos principales del mismo. Asimismo, se describen los diferentes escenarios que se han simulado y se resumen los resultados obtenidos. Finalmente, el apartado 5 contiene las conclusiones y trabajos futuros.

2 Gestión de la capacidad de los servicios TI

La finalidad principal del proceso *Gestión de la Capacidad* es proporcionar la capacidad necesaria para suministrar servicios de calidad a un coste razonable. Este proceso se compone de los siguientes subprocesos que analizan las necesidades de capacidad desde tres puntos de vista diferentes [10]: *Gestión de la Capacidad del Negocio* que prevee las necesidades futuras de los clientes, *Gestión de la Capacidad del Servicio* que analiza el rendimiento de los servicios TI con el objetivo de garantizar el cumplimiento de los acuerdos de nivel de servicios establecidos y *Gestión de la Capacidad de los Recursos* que analiza el uso de la infraestructura para asegurar que se dispone de los recursos necesarios y que se utilizan eficazmente.

En este trabajo nos centramos en la aplicación de los modelos dinámicos de simulación en el subproceso *Gestión de la Capacidad del Servicio*. En el contexto de este proceso, las políticas de gestión de la capacidad de los servicios que los proveedores asignan a sus clientes influyen en el cumplimiento de los SLAs.

En los siguientes apartados se presenta un modelo dinámico que simula el comportamiento de un servicio ante diferentes políticas de gestión de la capacidad y ayuda a determinar cuál de ellas es más adecuada respecto al cumplimiento de los SLAs. El modelo permite estudiar el rendimiento real del servicio con dos políticas de gestión de la capacidad diferentes y evaluar si el rendimiento es el acordado con el

cliente. Por otro lado, permite estudiar los costes que tendría que asumir el proveedor si no se cumplen los tiempos de respuesta.

3 Trabajos relacionados

La Tabla 1 [8] muestra las principales técnicas que se utilizan para resolver los problemas de gestión de servicios TI clasificadas en función de los tipos de problemas que ayudan a resolver.

Table 1. Principales técnicas de gestión de servicios TI

Técnicas	Tipo de problema
Inteligencia artificial	Toma de decisiones
Programación lineal, no lineal, entera y dinámica	Optimización y programación dinámica
Modelos probabilísticos	Estimación, predicción, inferencia y soporte a la decisión
Simulación	Predicción y soporte a la decisión

Las técnicas de *inteligencia artificial* y las técnicas de *programación lineal*, *no lineal*, *entera* y *dinámica* se suelen utilizar en situaciones deterministas. Los *modelos probabilísticos* se utilizan en situaciones inciertas. Los *modelos de simulación* son modelos computacionales que permiten representar de forma simplificada sistemas complejos. Estos modelos ofrecen como principal ventaja la posibilidad de experimentar diferentes decisiones y analizar sus resultados en sistemas donde el coste o el riesgo de una experimentación real son prohibitivos. Por otro lado, la simulación permite el análisis de sistemas de una complejidad tan elevada que resultan imposibles de representar mediante modelos analíticos. El objetivo común de los modelos de simulación es proporcionar mecanismos para la experimentación, predicción del comportamiento, resolución de preguntas del tipo *¿Qué pasaría si…?* y aprendizaje del sistema representado, entre otros. En este trabajo se utilizan modelos de simulación dinámicos continuos. Entre las principales ventajas de estos modelos frente a los modelos analíticos se encuentran las siguientes [14]:

- Constituyen un mecanismo flexible y útil para capturar y modelar el alto grado de incertidumbre que presentan algunos sistemas. Estos modelos complementan a las técnicas analíticas que permiten modelar el riesgo y el comportamiento asociado a dicha incertidumbre.
- Es una técnica flexible que permite representar un amplio rango de estructuras e interacciones dinámicas. Las técnicas analíticas pueden dar lugar a problemas intratables cuando la complejidad del sistema es elevada.
- Permiten modelar la realimentación de los sistemas. Hay sistemas en los que el comportamiento y decisiones tomadas en un instante determinado repercuten en la evolución del sistema. Cuando las implicaciones son complejas, los modelos analíticos son inaplicables o inútiles.

Entre los trabajos actuales que proponen aplicar estas técnicas en la gestión de servicios TI pueden encontrarse los siguientes: En [16] se utiliza el modelado de procesos y el análisis de la tecnología en la gestión de servicios TI. El ámbito de

aplicación de [15] es el proceso *Gestión de Cambios*. Se propone un modelo analítico de decisión basado en las técnicas de programación determinista y de programación probabilística que ayuda a planificar y a realizar los cambios en los servicios. El ámbito de aplicación de los trabajos [3] y [2] es el proceso *Gestión de Incidentes*. En [3] se propone un sistema de ayuda a la toma de decisiones dirigido por los objetivos del negocio. En [2] se presenta un enfoque de evaluación y mejora de la gestión de incidentes que se basa en una metodología de análisis de las causas del mal funcionamiento de los servicios que utiliza técnicas de minería de datos. Los trabajos [7] y [1] proponen utilizar las técnicas de simulación dinámica en la gestión de servicios TI. En [7] se propone un modelo que ayuda a gestionar los retrasos que se producen en un proceso sobre cuyas causas existe un cierto grado de incertidumbre. El modelo propuesto en [1] se aplica a la problemática de asignación de los recursos para garantizar el cumplimiento de los SLAs. En este trabajo se propone un modelo dinámico de simulación en el ámbito del proceso *Gestión de la Capacidad* de ITIL.

4 Caso de estudio

En este apartado se describe el caso de estudio en el que se enmarca el modelo dinámico de simulación propuesto y se explican las características del mismo. Asimismo, se describen los escenarios que se han simulado y se recogen y analizan los resultados obtenidos.

4.1 Descripción del Problema

Para el propósito de nuestro estudio hemos considerado una empresa proveedora de servicios de validación bancarios y una compañía de comercio electrónico que vende productos en Internet. En este contexto, la empresa proveedora de servicios y la compañía firmarán un SLA en el que se especifican acuerdos de nivel de servicio de diferentes categorías. El objetivo fundamental de este trabajo es evaluar cómo influye la gestión de la capacidad del servicio de validación de crédito que el proveedor asigna a la compañía, en el rendimiento del servicio y en el coste que tendría que asumir el proveedor si los tiempos de respuesta no son los acordados. Por ello, en este estudio se han considerado los parámetros del SLA que se suelen utilizar para definir estos aspectos [10]. Estos parámetros se han clasificado en las siguientes categorías:

- *Capacidad del servicio:*
  - *TasaValidaciónContratada*: capacidad del servicio contratada por la compañía.
- *Tiempos de respuesta:*
  - *TiempoRespuestaEsperado (TRE):* tiempo máximo que puede transcurrir entre una petición de validación y la respuesta del servicio sin que el proveedor del servicio sea penalizado.
  - *TiempoRespuestaMáximo (TRM):* tiempo máximo que puede transcurrir entre una petición de validación y la respuesta del servicio sin que la petición de validación sea abandonada.
- *Rendimiento del servicio:*

  - *TasaPeticionesValidadasTRE:* porcentaje mínimo de peticiones de validación que tienen que ser validadas en el tiempo de respuesta esperado.
  - *TasaPeticionesAbandonadas:* porcentaje máximo de peticiones de validación que se permite que se abandonen por sobrepasar el tiempo de respuesta máximo.
- *Penalizaciones por incumplimiento de los tiempos de respuesta:*
  - *PenalizaciónTiempoRespuestaEsperado:* penalización que tendría que asumir el proveedor del servicio por cada petición que se valida sobrepasando el tiempo de respuesta esperado.
  - *PenalizaciónAbandono:* penalización que tendría que asumir el proveedor del servicio por cada petición que se abandona por sobrepasar el tiempo de respuesta máximo.

En este trabajo se estudian los efectos de las siguientes políticas de gestión de la capacidad del servicio de validación de crédito en el cumplimiento del SLA:

- *Política A:* El proveedor del servicio establece el porcentaje de la tasa de validación de crédito que se utiliza para validar las peticiones en el tiempo de respuesta esperado. El resto de la tasa de validación de crédito se utiliza para validarlas en el tiempo de respuesta máximo. La tasa de validación de crédito que se asigna en un momento dado para validar las peticiones en los tiempos de respuesta establecidos siempre es la misma con independencia de la tendencia que presenten las peticiones recibidas.
- *Política B:* El proveedor del servicio establece el mayor porcentaje de la tasa de validación de crédito que se puede utilizar para validar las peticiones en el tiempo de respuesta máximo. La tasa de validación que no se utiliza para validar las peticiones en el tiempo de respuesta máximo, se utiliza para validarlas en el tiempo de respuesta esperado. A diferencia de la *Política A*, la tasa de validación de crédito que se asigna en un momento dado para validar las peticiones en los tiempos de respuesta establecidos varía en función de la tendencia que presenten las peticiones recibidas.

4.2 Construcción del modelo de simulación

Siguiendo la propuesta de Kellner para describir modelos de simulación [6] y la metodología de Martínez y Richardson [9], a continuación se describe el modelo de simulación construido. La implementación del modelo y las simulaciones se han realizado utilizando el entorno de simulación Vensim®.

4.2.1. Propósito y ámbito del modelo

El modelo propuesto se aplica a la problemática de analizar los efectos que las *Políticas A* y *B* de gestión de la tasa de validación de crédito tienen en el cumplimiento de los SLAs. El propósito del modelo es ayudar al proveedor del servicio a gestionar adecuadamente la tasa de validación de crédito para garantizar el cumplimiento de los parámetros de rendimiento del SLA. Asimismo, el modelo también permite estudiar los costes que tendría que asumir el proveedor del servicio por el incumplimiento de los tiempos de respuesta del SLA. El rendimiento real del servicio será el acordado siempre que:

- El porcentaje de peticiones que se validan en el tiempo de respuesta esperado sea mayor o igual al especificado en el parámetro *TasaPeticionesValidadasTRE* del SLA.
- El porcentaje de peticiones de validación que se abandonan por sobrepasar el tiempo de respuesta máximo sea menor o igual al especificado en el parámetro *TasaPeticionesAbandonadas* del SLA.

4.2.2. Variables de salida

Las principales variables que proporcionan información respecto del propósito del modelo son:

- *IncumplimientoTasaPeticionesValidadasTRE:* desviación que existe entre el parámetro *TasaPeticionesValidadasTRE* del SLA y la tasa de peticiones recibidas que se validan en el tiempo de respuesta esperado en un momento dado.
- *IncumplimientoTasaPeticionesAbandonadas:* desviación que existe entre la tasa de abandono de peticiones de validación recibidas que se produce en un momento dado por sobrepasar el tiempo de respuesta máximo y el parámetro *TasaPeticionesAbandonadas* del SLA.
- *CosteIncumplimientoTiemposRespuesta:* coste que tendría que asumir el proveedor por las peticiones validadas en el tiempo de respuesta máximo y por las peticiones abandonadas.

Otras variables de salida que son útiles para entender qué ocurre en el sistema son las siguientes:

- *PeticionesRecibidas:* número de peticiones de validación que recibe el proveedor del servicio.
- *PeticionesValidadasTRE:* número de peticiones validadas en el tiempo de respuesta esperado.
- *PeticionesValidadasTRM:* número de peticiones validadas en el tiempo de respuesta máximo.
- *PeticionesAbandonadas:* número de peticiones abandonadas.

4.2.3. Parámetros de entrada

Los parámetros de entrada permiten configurar diferentes escenarios de simulación. Los utilizados en este estudio se han clasificado de la siguiente manera:

- *Parámetros del SLA que han firmado la empresa proveedora del servicio y la compañía* (véase apartado 4.1)
- *Parámetros de gestión del servicio de validación:*
  - *TasaValidaciónCrédito:* capacidad de validación de crédito que el proveedor asigna a la compañía en un momento dado.
  - *PorcentajeTasaValidaciónCrédito:* porcentaje de la tasa de validación de crédito que el proveedor utiliza para validar las peticiones en el tiempo de respuesta esperado. El resto de la tasa de validación de crédito se utiliza para validarlas en el tiempo de respuesta máximo.
- *Parámetros que modelan la tendencia de las peticiones de validación recibidas:*
  - *TasaPeticionesRecibidas:* representa la tendencia de las peticiones de validación recibidas.

4.2.4. Proceso de abstracción

Para representar los elementos del modelo y sus relaciones se han utilizado los *diagramas causales*. Para modelar la componente de comportamiento del sistema y representar formalmente sus relaciones causa-efecto se han utilizado los *diagramas de flujo y niveles* o *diagramas de Forrester* [4]. Las *variables de nivel* representan las variables más importantes de un sistema o aquellas cuyo comportamiento se desea observar. Las *variables de flujo* representan cómo cambian las variables de nivel con respecto al tiempo y permiten modelar las políticas de operación o reglas de decisión de un sistema. Matemáticamente, las variables de nivel se modelan mediante ecuaciones diferenciales que integran a lo largo del tiempo la diferencia entre las variables de flujo de entrada y las variables de flujo de salida de dicha variable de nivel. El conjunto de ecuaciones resultantes constituye el modelo matemático del sistema que se resuelve a través de la simulación. En el modelo propuesto, las variables de salida *PeticionesValidadasTRE*, *PeticionesValidadasTRM* y *PeticionesAbandonadas* se han modelado como variables de nivel cuyo comportamiento se controla mediante las variables de flujo *TasaValidaciónTRE*, *TasaValidaciónTRM* y *TasaAbandono*. Las variables de salida *IncumplimientoTasaPeticionesValidadasTRE, IncumplimientoTasaPeticiones Abandonadas y CosteIncumplimientoTiemposRespuesta* se han modelado como variables auxiliares.

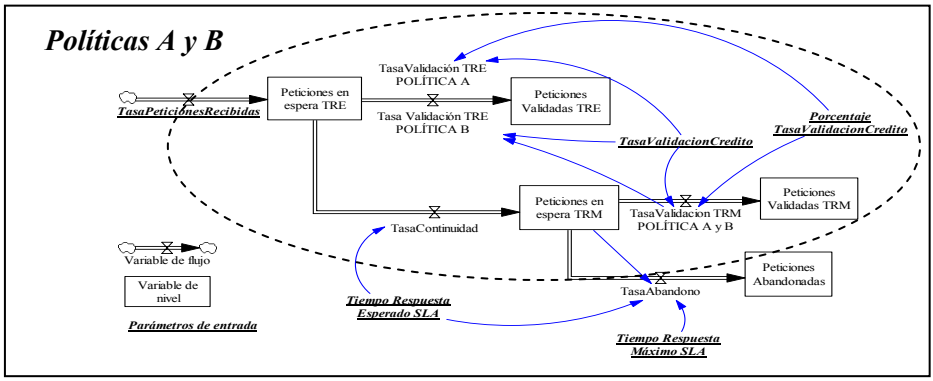


Fig. 1. Diagrama de flujos y niveles simplificado

En la Figura 1 se muestra el diagrama de flujos y niveles del modelo simplificado y se detallan los parámetros de entrada y variables que influyen en el cálculo de las variables de flujo *TasaValidaciónTRE* y *TasaValidaciónTRM* con las *Políticas A* y *B* de gestión de la tasa de validación de crédito.

4.3 Simulaciones del modelo

Las simulaciones del modelo permiten analizar los efectos de las *Políticas A* y *B* de gestión de la tasa de validación de crédito en el cumplimiento de los SLAs. A continuación se describe el caso de estudio considerado y se presentan y analizan los resultados obtenidos en los análisis de sensibilidad que se han realizado.

4.3.1 Configuración de parámetros de entrada para el caso de estudio

Las simulaciones del modelo se han realizado considerando que el proveedor del servicio y la compañía de comercio electrónico han firmado un determinado SLA y que las peticiones del servicio de validación presentan una tendencia concreta.

Parámetros del SLA que han firmado la empresa proveedora del servicio y la compañía:

Tomando como referencia ejemplos de SLAs de compañías de comercio electrónico, se han considerado los siguientes parámetros: *TasaValidaciónContratada:* 4000 peticiones/min. *TiempoRespuestaEsperado:* 15 segundos. *TiempoRespuestaMáximo:* 30 segundos. *TasaPeticionesValidadasTRE:* 90% de las peticiones recibidas. *TasaPeticionesAbandonadas:* 5% de las peticiones recibidas. *PenalizaciónTiempoRespuestaMáximo:* 0,2 euros/petición validada tiempo respuesta máximo. *PenalizaciónAbandono:* 1,8 euros/petición abandonada.

Parámetros que modelan la tendencia de las peticiones de validación

En la Figura 2 se muestra el parámetro de entrada *TasaPeticionesRecibidas* que representa la tendencia de las peticiones de validación que recibe el servidor. Se considera que la tasa de peticiones recibidas experimenta un incremento gradual hasta alcanzar un valor máximo que se mantiene constante durante un determinado periodo de tiempo. Pasado este periodo vuelve a descender gradualmente. El valor máximo de la tasa de peticiones recibidas considerado corresponde al record de ventas que registró Amazon en las navidades de 2008 [13].

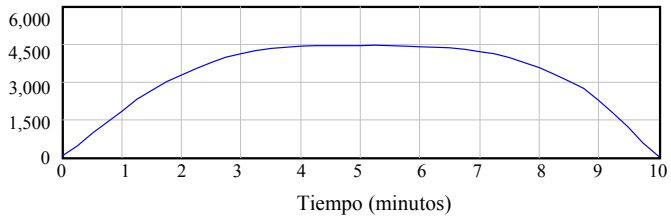


Fig. 2. Patrón de comportamiento del parámetro de entrada *TasaPeticionesRecibidas*.

4.3.2 Análisis de sensibilidad

Los objetivos de los análisis de sensibilidad realizados para determinar qué política de gestión de la tasa de validación de crédito es más adecuada respecto al cumplimiento de los SLAs son los siguientes:

- Estudiar los valores de los parámetros de gestión de la tasa de validación de crédito que garantizan el cumplimiento de los parámetros de rendimiento del SLA.
- Estudiar los costes que tendría que asumir el proveedor del servicio por el incumplimiento de los tiempos de respuesta del SLA.

Los análisis de sensibilidad se han realizado variando los parámetros de gestión del servicio de validación de crédito *TasaValidaciónCrédito* y *PorcentajeTasaValidaciónCrédito*. Los *parámetros del SLA* y la *TasaPeticionesRecibidas* se mantienen constantes (véase apartado 4.3.1).

4.3.2.1 Análisis de sensibilidad para estudiar el cumplimiento de los parámetros de rendimiento del SLA

La finalidad principal de estos análisis de sensibilidad es estudiar qué valores de la tasa de validación de crédito y de los porcentajes de la tasa de validación de crédito garantizan el cumplimiento de los parámetros de rendimiento del SLA. La configuración de parámetros de los análisis de sensibilidad se muestra en la Tabla 2.

Tabla 2. Configuración de parámetros de los análisis de sensibilidad para estudiar los parámetros de gestión

Número de simulaciones: 200
Parámetros de entrada: – *Parámetros del SLA y TendenciaPeticionesRecibidas:* valores caso de estudio (véase apartado 4.3.1).
Parámetros de control del análisis de sensibilidad: – *TasaValidaciónCrédito*: varía entre la *TasaValidaciónContratada* (4000 peticiones /minuto) y el valor mayor de la tendencia de peticiones de validación recibidas (4459 peticiones/minuto). **Distribución:** Uniforme aleatoria. – *PorcentajeTasaValidaciónCrédito:* varía entre 1% y 100%. **Distribución:** Uniforme aleatoria.

Los datos obtenidos en los análisis de sensibilidad indican que los valores de la *TasaValidaciónCrédito* y del *PorcentajeTasaValidaciónCrédito* que garantizan el cumplimiento de los parámetros de rendimiento del SLA son los que se muestran en la Tabla 3.

Tabla 3. *TasaValidaciónCrédito* y *PorcentajeTasaValidaciónCrédito* válidos (*Politicas A* y *B*)

TasaValidaciónCrédito	***PorcentajeTasa ValidaciónCrédito (Política A)***	***PorcentajeTasa ValidaciónCrédito (Política B)***
4239<=TVC<=4270	%TVC>=95%	%TVC>=95%
4271<=TVC<=4315	%TVC>=94%	%TVC>=94%
4316<=TVC<=4362	%TVC>=93%	%TVC>=93%
4363<=TVC<=4382	%TVC>=92%	%TVC>=92%
4383<=TVC<=4390	%TVC>=92%	%TVC>=91%
4390<=TVC<=4410	%TVC>=92%	
TVC>=4411	%TVC>=91%	
TVC>=4391		%TVC>=0%

4.3.2.2 Análisis de sensibilidad para estudiar los costes por el incumplimiento de los tiempos de respuesta del SLA

La finalidad de estos análisis de sensibilidad es estudiar los costes que tendría que asumir el proveedor del servicio por el incumplimiento de los tiempos de respuesta establecidos en el SLA en los escenarios en los que el rendimiento del servicio es el acordado. Para ilustrar este estudio, en este apartado se analizan los costes que tendría que abonar el proveedor cuando le asigna a la compañía alguna de las tasas de validación de crédito que garantizan el cumplimiento de los parámetros de rendimiento del SLA (véase Tabla 3). La Tabla 4 muestra la configuración de parámetros de los análisis de sensibilidad.

Tabla 4. Configuración de parámetros de los análisis de sensibilidad para estudiar los costes.

Número de simulaciones: 200
Parámetros de entrada: – *TasaValidaciónCrédito*: 4383 peticiones/minuto (obtenida de la Tabla 3) – *Parámetros del SLA y TendenciaPeticionesRecibidas:* valores caso de estudio (véase apartado 4.3.1)
Parámetros de control del análisis de sensibilidad: – *PorcentajeTasaValidaciónCrédito:* varía entre 92% y 100% con la *Politica A* y entre 91% y 100% en la *Política B* (obtenidos de la Tabla 3). **Distribución:** Uniforme aleatoria

La Figura 3 muestra la variable de salida *CosteIncumplimientoTiemposRespuesta* obtenida como resultado del análisis de sensibilidad realizado considerando la *Política A* (Figura 3a) y la *Política B* (Figura 3b).

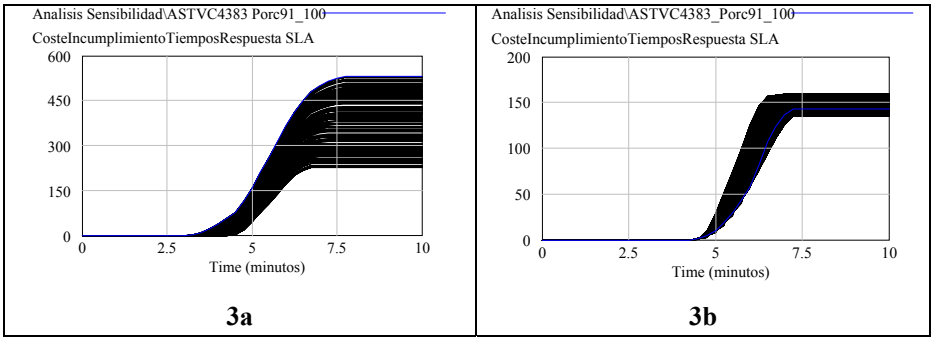


Fig. 3. Valores *CosteIncumplimientoTiemposRespuesta* con *Política A* (3a) y *Política B* (3b).

Del análisis de los datos obtenidos en los análisis de sensibilidad se concluye que la tendencia que presentan los costes por el incumplimiento de los tiempos de respuesta establecidos en el SLA, es diferente con las *Políticas A* y *B*:

– *Politica A:* cuanto mayor es el porcentaje de la tasa de validación de crédito asignado para validar las peticiones en el tiempo de respuesta esperado, más tarde se incumplen los tiempos de respuesta establecidos y el coste por incumplirlos es menor. Por tanto, el menor coste se produce utilizando el 100% de la tasa de validación de crédito para validar las peticiones en el tiempo de respuesta esperado.
– *Política B:* cuanto menor es el porcentaje de la tasa de validación de crédito asignado para validar las peticiones en el tiempo de respuesta esperado, los tiempos de respuesta se incumplen antes y el coste por incumplirlos es menor. Por tanto, el menor coste se produce asignado el 91% de la tasa de validación de crédito para validar las peticiones en el tiempo de respuesta esperado.

La característica dinámica del modelo utilizado permite no sólo determinar valores finales sino también analizar la evolución en el tiempo de las variables del sistema bajo estudio. En este caso, analizando la evolución en el tiempo de los costes por incumplimiento de los tiempos de respuesta con las *Políticas A* y *B* se observa que los costes se producen antes con la *Política A* que con la *Politica B*. Asimismo, se observa que los costes con la *Política A* son mayores que con la *Política B*.

5 Conclusiones y trabajos futuros

En este trabajo se presentan un conjunto de los resultados de una iniciativa de investigación que se centra en la utilización de las técnicas de modelado y simulación en los procesos de gestión de servicios TI. En el ámbito del proceso Gestión de la Capacidad de ITIL, se ha desarrollado un modelo dinámico de simulación que ayuda a analizar los efectos que las políticas de gestión de la capacidad de los servicios que los proveedores asignan a sus clientes tienen en el cumplimiento de los SLAs. En concreto, ayuda a decidir cuál de las políticas de gestión descritas en el apartado 4.1 (*Políticas A* y *B*) es más adecuada respecto al cumplimiento de los acuerdos de nivel de servicio establecidos. Para ello, el modelo permite estudiar los efectos que estas políticas tienen sobre el rendimiento de los servicios y sobre los costes que tendría que asumir el proveedor si los tiempos de respuesta no son los acordados. Los análisis de sensibilidad realizados se han configurado variando la capacidad del servicio, la forma de gestionarla y los porcentajes de esta capacidad que se utilizan para responder las invocaciones del servicio recibidas por parte del cliente en los tiempos de respuestas acordados con el proveedor. La finalidad concreta de estos análisis de sensibilidad es evaluar si la capacidad del servicio contratada por un cliente es suficiente para garantizar el cumplimiento de los parámetros de rendimiento del SLA. Asimismo, ayudan a detectar las situaciones en las que capacidades inferiores al número de invocaciones al servicio recibidas por parte de un cliente garantizan que el rendimiento del servicio es el acordado. Por otro lado, permiten evaluar los costes que se producen en estas situaciones por el incumplimiento de los tiempos de respuesta establecidos en el SLA.

En el caso de estudio considerado en este trabajo, se obtienen las siguientes conclusiones:

- Con las *Políticas A* y *B*, el rendimiento del servicio con la capacidad contratada por el cliente es inferior al acordado en el SLA.
- Las situaciones en las que capacidades del servicio inferiores al número de invocaciones al servicio recibidas por parte del cliente garantizan que el rendimiento del servicio es el acordado, son diferentes con las *Políticas A* y *B*.
- Para una determinada capacidad, el rendimiento del servicio con la *Política B* es mejor que con la *Política A*. Asimismo, los costes por incumplir los tiempos de respuesta establecidos, son menores y tardan más en producirse con la *Política B* que con la *Política A*.

La finalidad principal de nuestros próximos trabajos es la siguiente:

- Estudiar los efectos que diferentes valores de los parámetros del SLA tienen sobre el rendimiento de los servicios y sobre los costes que tendrían que asumir los proveedores si el rendimiento no es el acordado con el cliente. La característica dinámica del modelo propuesto permite no sólo determinar los valores finales de estas variables sino también analizar su evolución en el tiempo. El proveedor del servicio podría utilizar los resultados de este estudio como base de la renegociación del SLA que ha firmado con el cliente.
- Ampliar el modelo incluyendo otras políticas de gestión de la capacidad diferentes a las estudiadas en este trabajo.

Agradecimientos

Esta investigación está parcialmente financiada por el Ministerio de Educación y Ciencia de España y por los fondos europeos FEDER mediante el proyecto TIN2007-67843-C06-04.

Referencias

1. An, L.,Jeng,J.J.: Web Services Management Using System Dynamics. En Proceedings of the IEEE International Conference on Web Services (ICWS'05), pp.347-354 (2005).
2. Barash., G. Bartolini, C., Liya, W.: Measuring and Improving the Performance of an IT Support Organization in Managing Service Incidents. En Business-Driven IT Management (BDIM'07), pp.11-18 (2007).
3. Bartolini, C., Sallé, M., Trastour, D.: IT Service Management Driven by Business Objectives : An Application to Incident Management. En 10th IEEE/IFIP Network Operations and Management Symposium (NOMS) (2006).
4. Forrester, J.W: Industrial Dynamics. Pegaus Communications. Massachussets (1961).
5. itSMF International: Fundamentos de Gestión de Servicios TI basado en ITIL. Van Haren Publishing (2008).
6. Kellner, M.I., Madachy, R.J., Raffo, D.: Software process simulation modeling: Why? What? How? J. Syst. Software, 46(2-3): 91-105 (1999)
7. Lee J.H., Han, Y.S., Kum C.H.: IT Service Management Case based Simulation Analysis & Design: Systems Dynamics Approach. En IEEE International Conference on Convergence Information Technology, pp 1559-1566. IEEE Computer Society, Washington, DC (2007).
8. Liu, D., Deters,R.: Management Service-Oriented Systems. En: Springer London (eds). Service Oriented Computing and Applications 2(2-3). pp 51-64 (2008).
9. Martínez, I.J., Richardson, G.P.: Best Practices in System Dynamics Modeling. Proceedings of the 19th International Conference of the Systems Dynamics Society. Atlanta, USA (2001).
10. Office of Government Commerce: Service Design. The Stationary Office (TSO) (2007).
11. Office of Government Commerce: The Official Introduction to the ITIL Service Lefecycicle. The Stationary Office (TSO) (2007).
12. Portal oficial de ITIL http://www.itil-officialsite.com.
13. Portal oficial de El Economista http:// www.eleconomista.es/empresas-finanzas/noticias/942024/12/08/-Amazoncom-registra-su-mejor-temporada-de-ventas-navidenas.html.
14. Ruiz, M., Ramos, I., Toro, M.: Modelado y Simulación del proceso de desarrollo de software: Una técnica para la mejora de procesos. En: CEUR Workshop Proceedings Vol.84 (2001). http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-84/.
15. Setzer, T. , Bhattacharya, K.,: Decision Support for Service Transition Management Enforce Change Scheduling by Performing Change Risk and Business Impact Analysis. En Network Operations and Management Symposium, 2008. NOMS 2008. IEEE. pp.200-207 (2008).
16. Shen, B: Support IT Service Management with Process Modeling and Analysis. En Q.Wang, D.Pfahl, and D.M. Raffo (eds.): JCSP 2008, LNCS 5007, pp.246-256. Springer-Verlag Berlin Heidelberg (2008).

A tabu search algorithm for structural software testing

Eugenia Díaz[1], Javier Tuya[1], Raquel Blanco[1], José Javier Dolado[2]

[1] Department of Computer Science, University of Oviedo, Campus de Viesques, Gijón, Asturias 33204, Spain
{madiaz, tuya, rblanco}@uniovi.es
[2] Department of Computer Science and Languages, University of the Basque Country, Spain
javier.dolado@ehu.es

Abstract. This paper presents a tabu search metaheuristic algorithm for the automatic generation test cases using a branch coverage adequacy criterion. It is a novel work since tabu search is applied to the automation of the test generation task, whereas previous works have used other techniques such as genetic algorithms.

The developed test generator, called TSGen, generates a set of neighboring test candidates based on a test that is selected according to which subgoal branch has to be covered. Then, each test is checked in order to determine whether it is a tabu test. In that case, the tabu test is stored in a TSGen memory (tabu list) and it is rejected. If a test candidate is not tabu, the program under test is executed to check the branches that have been covered and the cost incurred by said test. TSGen has a cost function for intensifying the search and another for diversifying the search that is used when the intensification is not successful. During the search process, the best solutions found are stored together with their costs in order to use them in the generation of the new tests. TSGen also combines the use of memory with a backtracking process to avoid getting stuck in local minima.

Evaluation of the generator was performed using benchmarks that have branches for which it is very difficult to find a test that reaches them because of the existence of several of the factors that influence the complexity of reaching a branch (high nesting complexity, high condition complexity, the use of equality and remainder operators in conditions and the use of AND operators in decisions). In spite of the existence of these difficulties, TSGen achieves 100% branch coverage for all the benchmarks and consumes a reasonable time to do so. Moreover, the performance of TSGen is highly independent with regard to the range of input variables.

Abstract of the paper included in the Computers and Operations Research Journal, Vol. 35, Issue 10, pp 3052-3072, 2008.

WebAVLTester: Una Herramienta de Pruebas Funcionales Automáticas para Formularios Web

Eugenia Díaz, Marta Fernández de Arriba, Roberto López

Departmento de Informática, Universidad de Oviedo
Campus de Viesques, Gijón, Asturias, 33204, España
{madiaz, martafer}@uniovi.es, roberto.lopez@arcelor.com

Resumen. La prueba de sitios Web es una tarea tediosa y costosa que consume mucho tiempo y que es necesaria para obtener sitios Web de calidad. Existen diversas herramientas para automatizar parte de las pruebas que deberían de hacerse a un sitio Web, incluyendo herramientas para grabar y reproducir casos de prueba facilitando la prueba de regresión. Sin embargo, la creación de casos de prueba funcionales es un proceso manual que requiere un probador experimentado. Este artículo presenta una herramienta, denominada WebAVLTester, que automatiza el proceso de creación de casos de prueba para formularios Web. Para ello, utiliza el código XHTML para guiar al probador en la introducción de restricciones de prueba y genera automáticamente los casos de prueba funcionales empleando la técnica de valores límite, permitiendo su prueba automática y registrando el resultado obtenido.

Palabras clave: Herramienta de pruebas Web, Pruebas funcionales, Análisis de Valores Límite, Generación automática de casos de prueba.

1 Introducción

La prueba del software es un proceso caro que suele consumir al menos un 50% del coste total del desarrollo software [1]. En concreto, la prueba de sitios Web implica emplear mucho tiempo para diseñar los diferentes tipos de pruebas que pueden realizarse (funcionalidad, usabilidad, accesibilidad, rendimiento, seguridad, etc.).

Para reducir el coste del proceso de prueba de sitios Web, se pueden emplear una serie de herramientas que permiten automatizar algunos tipos de pruebas, como por ejemplo validadores sintácticos [2], comprobadores de enlaces [3], analizadores de accesibilidad [4] y de rendimiento [5]. También, existen herramientas que permiten automatizar partes de la prueba funcional, como por ejemplo WinRunner [6].

Sin embargo, la creación de casos de prueba funcionales (o de caja negra) es un proceso artesanal que requiere, además de diseñar manualmente los casos de prueba, su inyección manual en los formularios Web objetos de la prueba. Como alternativa a la inyección manual se pueden codificar un conjunto de guiones de prueba específicos (test scripts) disminuyendo el tiempo de prueba pero, incrementando la dificultad de crear buenos casos de prueba (debido a la especialización requerida para crear los test scripts) y aumentando la probabilidad de cometer errores en su propia creación.

La prueba funcional de sitios Web es esencial cuando existen formularios ya que éstos permiten el envío de datos erróneos, bien sea por la inexperiencia de los usuarios o por fines malintencionados. Por tanto, la existencia de formularios Web implica que la prueba funcional será fundamental para asegurar la calidad del sitio Web desarrollado. Además, debido a su realización principalmente de manera manual, la prueba funcional de formularios Web es una tarea muy tediosa y costosa de la fase de pruebas del software y, proclive a la existencia de errores humanos. Estos inconvenientes serían evitables si el proceso de diseño y ejecución de casos de prueba estuviese muy automatizado siendo éste, precisamente, el fin de la herramienta que presentamos en este artículo y que hemos denominado WebAVLTester (implementada en Java con el entorno de desarrollo ECLIPSE y la librería JTidy).

2 Visión general de WebAVLTester

WebAVLTester genera automáticamente casos de prueba funcionales (válidos y no válidos) para formularios Web escritos en XHTML [7]. Nuestra herramienta **evita que el probador tenga que escribir manualmente los casos de prueba o los test scripts** al proporcionar un interfaz guiado dónde tan sólo tiene que introducir las restricciones adecuadas conforme le sean solicitadas.

Es más, WebAVLTester utiliza la especificación del formulario XHTML y las restricciones del probador para generar automáticamente casos de prueba **utilizando el método del Análisis de Valores Límite** (AVL), basado en el método de partición en categorías [8], en el cuál los casos de prueba seleccionados son aquellos que están en los límites de las particiones al tener una mayor probabilidad de encontrar errores.

Además, nuestra herramienta **permite inyectar automáticamente los casos de prueba generados** en el formulario bajo prueba **y almacenarlos junto con el resultado obtenido para una posterior ejecución** cuándo el probador lo desee. Esto **facilitará la realización de pruebas de regresión**.

Por último, WebAVLTester permite especificar la salida esperada (oracle) que se obtendría, especificando el tipo de página (correcta o incorrecta) a devolver por el servidor en función de los datos introducidos. De esta manera, también **automatiza la comprobación de la salida esperada para todos los casos de prueba generados**.

La figura 1 muestra el esquema general de funcionamiento de nuestra herramienta, cuyo proceso de funcionamiento consta de los siguientes pasos:

1. Por medio del navegador que incluye, se captura el formulario Web que se desea probar. WebAVLTester extrae todos los campos del formulario capturado y sus propiedades. Este paso incluye los cinco primeros flujos de datos mostrados en la figura 1.
2. WebAVLTester guía al probador para introducir las diferentes restricciones aplicables a cada campo del formulario a probar. Debido a las numerosas posibilidades que nuestra herramienta ofrece según el tipo de campo bajo prueba, este paso está descrito con más detalle en la sección 3. Este paso incluye el flujo de datos número 6 indicado en la figura 1.
3. Utilizando el botón “Generar”, WebAVLTester generará automáticamente (aplicando AVL) los casos de prueba apropiados para probar el formulario

Web de acuerdo a las restricciones del probador. Este paso se corresponde con los flujos de datos 7, 8 y 9 de la figura 1.

4. A continuación, nuestra herramienta permite inyectarlos directamente en el formulario bajo prueba y enviarlos al servidor Web dónde se aloje dicho formulario. Este paso incluye los flujos de datos 10 a 14 de la figura 1.
5. Además, WebAVLTester permite generar numerosos informes de prueba como parte de la documentación necesaria que debe ir asociada al proceso de prueba. Este paso se corresponde con los flujos de datos 15 y 16 mostrados en la figura 1.

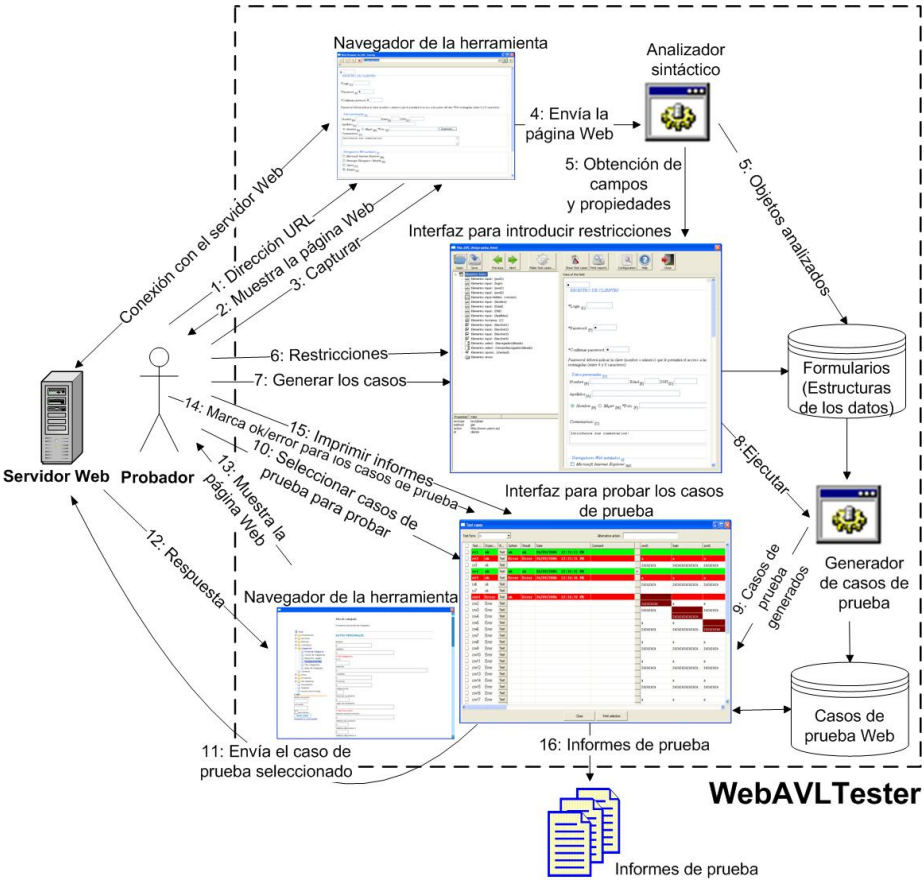


Fig. 1. Esquema general de WebAVLTester.

3 Detalle de la introducción de restricciones en WebAVLTester

Una vez capturado el formulario bajo prueba, WebAVLTester muestra un interfaz como el ejemplo de la Figura 2. En la parte izquierda superior aparecen todos los

campos del formulario indicando de qué tipo son. Al seleccionar un campo, en la parte izquierda inferior se muestran las restricciones obtenidas de la especificación XHTML. En la parte derecha y en función del tipo de campo seleccionado el probador podrá introducir las restricciones. En el ejemplo, se puede indicar si el campo será o no obligatorio, el tipo de datos, la longitud así como valores permitidos.

WebAVLTester comprueba las restricciones introducidas por el probador evitando que introduzca restricciones incoherentes entre sí o contradictorias con respecto a las propiedades descritas en el código del formulario.

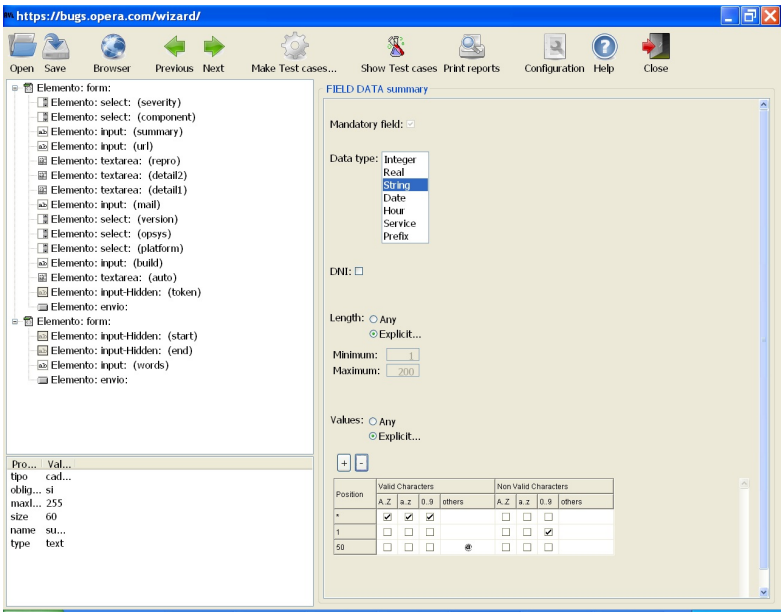


Fig. 2. Interfaz para la introducción de restricciones de prueba.

Agradecimientos. Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia de España dentro del Plan Nacional de I+D+I, Proyecto TIN2007-67843-C06-01.

Referencias

1. Beizer, B., Software Testing Techniques, 2nd. Ed. Van Nostrand Reinhold. 1990.
2. W3C Validator Team: W3C Markup Validation Service. URL: http://validator.w3.org
3. W3C Validator Team: W3C Link Checker. URL: http://validator.w3.org/checklink
4. TAW: Test de Accesibilidad Web. URL: http://www.tawdis.net/taw3/cms/es
5. AutomatedQA: Testcomplete™. URL: http://www.automatedqa.com/products/testcomplete/
6. Mercury Interactive: WinRunner©. http://www.powertest.com/sub2.pages/winrunner.html
7. W3C, Pemberton, S. et al.: XHTML 1.0: The Extensible HyperText Markup Language (Second Edition). A Reformulation of HTML 4 in XML 1.0.
8. Ostrand, T.J., Balcer, M.J.: The category partition method for specifying and generating functional tests. In Communications of the ACM, Vol. 31(6) (1988) 676–686.

GAmera: una herramienta para la generación y selección mediante algoritmos genéticos de mutantes WS-BPEL

A. Estero Botaro, I. Medina Bulo, J. J. Domínguez Jiménez y L. Gutiérrez Madroñal

Dpto. Lenguajes y Sistemas Informáticos, Universidad de Cádiz C/ Chile, N.º 1, 11003, Cádiz
{antonia.estero,inmaculada.medina,juanjose.dominguez,
lorena.gutierrez}@uca.es

Resumen En este artículo se presenta una herramienta novedosa, GAmera, para la generación y ejecución automática de mutantes para composiciones de servicios web en WS-BPEL. GAmera incorpora un mecanismo de optimización que permite seleccionar un subconjunto de los mutantes totales que pueden generarse. Esto se logra mediante la utilización de un algoritmo genético que genera y selecciona sólo los mutantes de mayor calidad, reduciendo el coste computacional que implicaría la ejecución de todos los mutantes. Los resultados que proporciona esta herramienta permiten mejorar la calidad de los casos de prueba.

1. Introducción

La evolución del software hacia las arquitecturas orientadas a servicio ha conducido a la definición de un lenguaje que facilite la composición de servicios web, el estándar OASIS WS-BPEL 2.0 [1], que se ha convertido en la referencia a nivel industrial. WS-BPEL permite especificar la lógica de la composición de los servicios (envío de mensajes, sincronización, iteración, tratamiento de transacciones erróneas, etc.) independientemente de su implementación.

La prueba de mutaciones ha sido validada como una poderosa herramienta para la evaluación de la calidad de los casos de prueba. Para poder aplicarla a las composiciones WS-BPEL es necesario definir un conjunto de operadores de mutación para este lenguaje y construir una herramienta que genere y ejecute los mutantes generados, con objeto de poder automatizar el proceso de pruebas.

Rice [2] enumera y explica los diez retos más importantes en la automatización del proceso de pruebas. Entre ellos, se encuentra la falta de herramientas, bien por su elevado precio o bien porque las existentes no se ajusten al propósito o entorno para el que se necesitan. Existen varios trabajos en la bibliografía que abordan el desarrollo de sistemas para la generación automática de mutantes, tales como Mothra [3] para Fortran, MuJava [4] para Java, Proteum [5] para C, y SQLMutation [6] para SQL. Todos estos sistemas se caracterizan por generar todos los mutantes posibles de acuerdo a los operadores de mutación empleados.

Uno de los principales inconvenientes que presenta la prueba de mutaciones es el alto coste computacional que supone la ejecución de la gran cantidad de mutantes que se generan para el programa de prueba. Dado que ésta es la primera herramienta que

Actas JISBD 2009, pp.50-53, ISBN: 978-84-692-4211-7

se construye para la aplicación de pruebas de mutaciones a composiciones WS-BPEL, hemos considerado que sería interesante introducir en ella un mecanismo de optimización que permitiera seleccionar sólo un subconjunto de los mutantes totales que pueden generarse. Para ello GAmera incorpora un algoritmo genético que selecciona sólo los mutantes de mayor calidad, reduciendo el coste computacional que supondría la ejecución de todos los mutantes.

En [7] presentamos un conjunto de operadores de mutación específicos para el lenguaje WS-BPEL 2.0 y en [8] un *framework* para la generación automática de mutantes para WS-BPEL 2.0 basado en algoritmos genéticos. La herramienta que se presenta en este artículo es fruto de estos trabajos previos. Este artículo tiene como objetivo mostrar la funcionalidad y utilidad de la herramienta GAmera, que implementa el algoritmo genético integrado con los operadores de mutación definidos para WS-BPEL en los trabajos previos. Esta herramienta se ha desarrollado como software libre, y puede descargarse gratuitamente de su web oficial [9].

2. Estructura de GAmera

La herramienta GAmera está constituida por tres componentes principales: el analizador, el generador de mutantes y el sistema de ejecución, que ejecuta y evalúa los mutantes.

El analizador Es el primer componente de la herramienta que actúa. Recibe como entrada la composición WS-BPEL a probar y determina los operadores de mutación que se le pueden aplicar. La figura 1 muestra la pantalla de la aplicación que interactúa con el analizador.

Los operadores se muestran en pestañas independientes, según la categoría del operador, permitiéndose al usuario incluso poder determinar el conjunto de operadores a utilizar de los que son aplicables a la composición WS-BPEL con la que se está trabajando.

Generador de mutantes Los mutantes se generan a partir de la información que se recibe del analizador. La herramienta nos da la posibilidad de generar todos los mutantes posibles, o bien, un subconjunto de éstos que va a ser seleccionado por el algoritmo genético. En este último caso, se llamará al componente denominado Generador genético de mutantes, que está compuesto por dos elementos. El primero, denominado Búsqueda Genética de Mutantes, es un algoritmo genético en el que cada individuo representa a un mutante, capaz de generar y seleccionar de forma automática un conjunto de mutantes. Esta selección se realiza aplicando una función de aptitud que mide su calidad en función de si hay o no casos de prueba que lo matan [8].

El segundo elemento es el Conversor, que transforma un individuo del algoritmo genético en un mutante WS-BPEL. Para realizar esta conversión, se utilizan hojas de estilos XSLT, una por cada operador de mutación.

Ejecución de mutantes A medida que se van generando los mutantes, el sistema los ejecuta frente a un conjunto de casos de prueba, distinguiéndose tres posibles estados para cada mutante según la salida que producen.

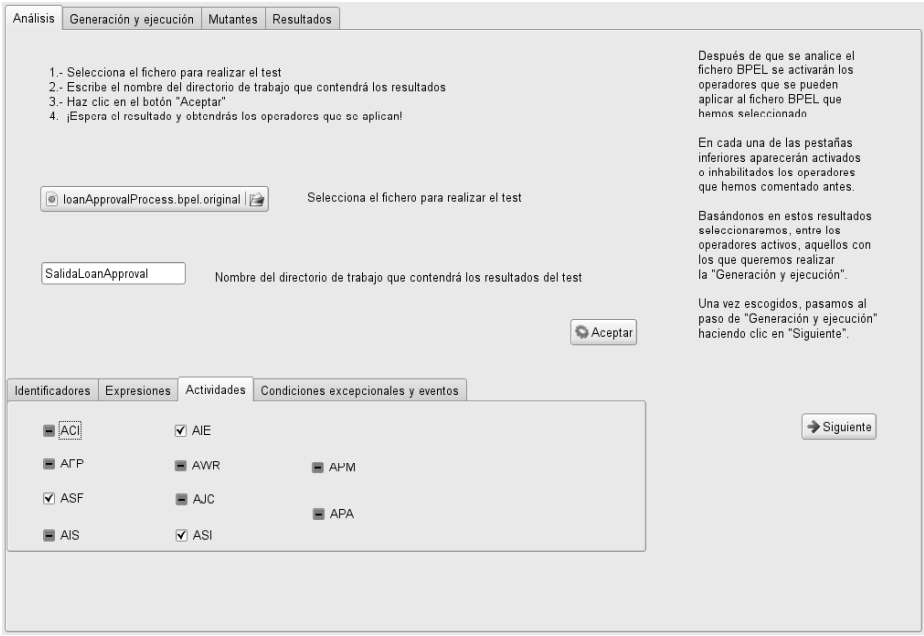


Figura 1. Interacción con el analizador

Muerto La salida del mutante es diferente a la del proceso original para al menos un caso de prueba.

Vivo La salida del mutante es la misma que la del proceso original para todos los casos de pruebas suministrados.

Erróneo Se ha producido un error en el despliegue del mutante y no se ha podido ejecutar. La existencia de este estado permite determinar si el diseño e implementación de los operadores de mutación es adecuado, o bien, si se están generando mutantes que no se pueden desplegar.

Para la ejecución del programa original y los mutantes, GAmera emplea el motor WS-BPEL 2.0 ActiveBPEL 4.1 [10] y BPELUnit [11], una biblioteca de pruebas unitarias para WS-BPEL que utiliza ficheros XML para describir los casos de prueba.

2.1. Resultados

GAmera permite visualizar los resultados obtenidos en la ejecución de los mutantes. Muestra el número total de mutantes generados, el de mutantes muertos, vivos y erróneos. También muestra estos valores para cada operador de mutación utilizado. A partir de estos valores se puede medir la calidad del conjunto de casos de prueba utilizado.

3. Conclusiones

Hemos presentado una herramienta que genera automáticamente un conjunto de mutantes de calidad para composiciones de servicios WS-BPEL mediante un algoritmo genético. La herramienta permite automatizar la prueba de composiciones WS-BPEL y proporciona información que permite mejorar la calidad de los conjuntos de casos de prueba.

La herramienta es de utilidad tanto para desarrolladores de composiciones WS-BPEL, ya que automatiza el proceso de prueba, como para investigadores en pruebas de composiciones WS-BPEL que tengan como finalidad evaluar la calidad de conjuntos de casos de prueba.

Actualmente estamos trabajando en el diseño de nuevos operadores de mutación para WS-BPEL 2.0 que contemplen diversos criterios de cobertura.

4. Agradecimientos

Este trabajo ha sido financiado por el Programa Nacional de I+D+I del Ministerio de Educación y Ciencia y fondos FEDER mediante el proyecto SOAQSim (TIN2007-67843-C06-04).

Referencias

1. OASIS: Web services business process execution language 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (2007)
2. Rice, R.: Surviving the top 10 challenges of software test automation. CrossTalk: The Journal of Defense Software Engineering (2002) 26–29
3. King, K.N., Offutt, A.J.: A fortran language system for mutation-based software testing. Software - Practice and Experience **21**(7) (1991) 685–718
4. Ma, Y.S., Offutt, J., Kwon, Y.R.: Mujava: an automated class mutation system. Software Testing, Verification & Reliability **15**(2) (2005) 97–133
5. Delamaro, M., Maldonado, J.: Proteum–a tool for the assessment of test adequacy for c programs. En: Proceedings of the Conference on Performability in Computing System. (1996) 79–95
6. Tuya, J., Cabal, M.J.S., de la Riva, C.: Mutating database queries. Information and Software Technology **49**(4) (2007) 398–417
7. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I.: Mutation operators for WS-BPEL 2.0. En: ICSSEA 2008: Proceedings of the 21th International Conference on Software & Systems Engineering and their Applications. (2008)
8. Domínguez-Jiménez, J.J., Estero-Botaro, A., Medina-Bulo, I.: A framework for mutant genetic generation for WS-BPEL. En: SOFSEM 2009: Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science, Lecture Notes in Computer Science. Volume 5404., Springer-Verlag (2009) 229–240
9. Grupo SPI&FM: Web oficial de GAmera. http://neptuno.uca.es/˜gamera
10. ActiveVOS.: Activebpel WS-BPEL and BPEL4WS engine. http://sourceforge.net/projects/activebpel (2008)
11. Mayer, P., Lübke, D.: Towards a bpel unit testing framework. En: TAV-WEB '06: Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications, ACM (2006) 33–42

Parte III

Sesión 2. Gestión de Proyectos

TUNE-UP: Seguimiento de proyectos software dirigido por la gestión de tiempos

María Isabel Marante[1], Patricio Letelier[1] y Francisco Suárez[2]

[1] Departamento Sistemas Informáticos, Universidad Politécnica de Valencia
[2] Aphelion Soluciones Informáticas S.L.
{ mmarante, letelier }@dsic.upv.es, Francisco.Suarez@addinformatica.com

Resumen. El convencimiento de que el proceso de desarrollo de software es determinante para el éxito de un proyecto ha impulsado el interés por estándares para la evaluación y mejora de procesos, junto con la implantación de metodologías. Sin embargo, ni los estándares para proceso software ni las metodologías modernas más populares han conseguido satisfacer totalmente las expectativas de equipos de desarrollo, particularmente en lo relativo a conseguir un mayor control en sus proyectos. TUNE-UP es una metodología nacida de las necesidades de mejora de proceso en una PYME de desarrollo de software. TUNE-UP combina aspectos de metodologías ágiles y tradicionales, pero su característica más innovadora es el énfasis en la gestión de tiempos como factor clave para la planificación y seguimiento del proyecto. Este trabajo ilustra el enfoque ofrecido por TUNE-UP para gestión de tiempos en proyectos software.

Palabras clave: Proceso software, metodologías para desarrollo de software, planificación de proyectos software.

1 Introducción

Un proyecto de desarrollo y/o mantenimiento de software conlleva todas las dificultades de un proyecto de ingeniería, pero además incluye los particulares retos que tiene la construcción de un producto software; complejidad de la implementación, requisitos volátiles, desafíos tecnológicos, desarrollo colaborativo, dificultad para asegurar la calidad, etc. Además, el mercado cada vez exige plazos de entrega más reducidos y presupuestos más ajustados.

En la última década hemos visto un creciente interés por metodologías y estándares asociados al proceso de desarrollo de software. El modelo de referencia CMMI [2] y los estándares SPICE [9] e ISO 90003 [10] han logrado reconocimiento y atraído el interés de empresas desarrolladoras de software. Metodologías Ágiles (p.e. XP [1] y Scrum [16]) y Tradicionales (p.e. RUP [11] y Métrica 3 [12]) han animado una interesante discusión respecto de las estrategias para desarrollar software y su efectividad en diferentes contextos, confirmándose que cada proyecto requiere un proceso ajustado a sus necesidades, es decir, deben considerarse: composición del equipo, envergadura del proyecto, características del dominio de aplicación, tecnología utilizada, etc.

Los proyectos de desarrollo de software suelen enfrentarse a un ámbito de trabajo muy cambiante (especialmente en cuanto a las especificaciones del producto y las prioridades de las características solicitadas) e intensivo en comunicación (entre el equipo y con el cliente). Las técnicas y herramientas genéricas para seguimiento de proyectos resultan claramente ineficaces para enfrentar estos desafíos. Las metodologías ágiles destacan esta situación pero la resuelven de una manera excesivamente simplista, utilizando roles muy genéricos (reduciendo así la comunicación necesaria entre diferentes roles) y confiando en la habilidad de cada uno de los miembros del equipo para que, verbalmente y/o con soportes no informatizados [7], realicen el seguimiento continuo del proyecto.

TUNE-UP es una metodología nacida en el trabajo día a día en una PYME de desarrollo de software y con una vocación de mejora continua del proceso. En tres años de aplicación y evolución TUNE-UP ha conseguido una madurez suficiente para ser presentada como una alternativa interesante, al menos en contextos de desarrollo con equipos pequeños. TUNE-UP incorpora elementos de metodologías ágiles y también del ámbito más tradicional. Una de las características clave de TUNE-UP es la planificación y seguimiento del proyecto centrada en la gestión de tiempos. TUNE-UP se inspira en la esencia de PSP (Personal Software Process) [17], donde se destaca que la base del éxito radica en una disciplina de trabajo y productividad individual centrada en la gestión de los compromisos. TUNE-UP ayuda en cada momento del proyecto a responder a la pregunta: ¿conseguiré cumplir con los plazos de entrega de mis tareas? o ¿seremos capaces de cumplir con los plazos de entrega al cliente? Estas simples preguntas inquietan a cualquier gestor o participante de un proyecto. No contar con una respuesta acertada y sobre todo oportuna conlleva en la mayoría de los casos graves complicaciones en el proyecto.

El objetivo de este trabajo es ilustrar cómo TUNE-UP aborda la gestión de tiempos contribuyendo a que todos los agentes, y en especial el Product Manager (jefe de desarrollo y mantenimiento de un producto software), tengan mayor control respecto de los plazos y compromisos en un proyecto de desarrollo o mantenimiento de software.

En la sección siguiente se describe brevemente la metodología TUNE-UP. En la sección 3 se detalla la gestión de tiempos en TUNE-UP. Posteriormente en la sección 4 se presenta TUNE-UP Process Tool, una herramienta de apoyo al proceso. En la sección 5, se comentan los trabajos relacionados. Finalmente, en la sección 6 se establecen las conclusiones.

2 Introducción a TUNE-UP

TUNE-UP es una metodología que incorpora aspectos ágiles y tradicionales con un sentido marcadamente pragmático. TUNE-UP se caracteriza fundamentalmente por combinar los siguientes elementos:

- **Modelo iterativo e incremental** para el desarrollo y mantenimiento del software. El trabajo se divide en unidades de trabajo que son asignadas a versiones del producto. Se realizan ciclos cortos de desarrollo, entre 3 y 6 semanas, dependiendo del producto.

- **Workflows flexibles** para la coordinación del trabajo asociado a cada unidad de trabajo. Los productos, según sus características, tienen disponible un conjunto de workflows los cuales se asignan a cada una de las unidades de trabajo.
- **Proceso de desarrollo dirigido por las pruebas de aceptación**. La definición de una unidad de trabajo es básicamente la especificación de sus pruebas de aceptación. A partir de allí, todo gira en torno a ellas, se estima el esfuerzo de implementar, diseñar y aplicar dichas pruebas, se diseñan e implementan y luego se aplican para garantizar el éxito de la implementación.
- **Planificación y seguimiento centrados en la gestión del tiempo**. Este artículo se centra precisamente en este aspecto.

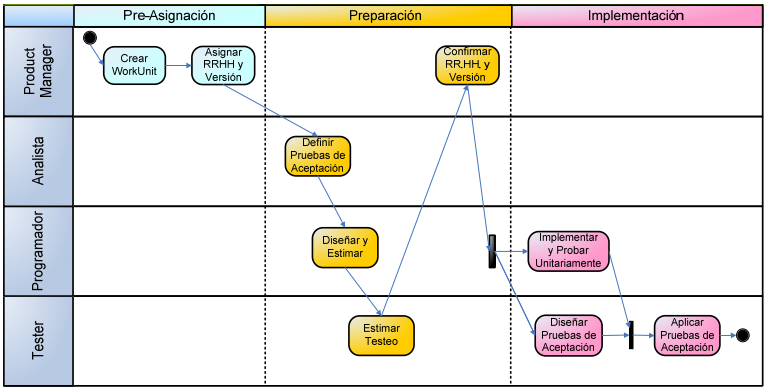


Fig. 1. Un workflow de desarrollo simple para unidades de trabajo

La Fig.1 ilustra un workflow mínimo para el desarrollo de una unidad de trabajo. Un workflow, en general, incluye actividades asociadas a tres fases del proceso:

- **Pre-Asignación**: actividades hasta la asignación de los RRHH y versión.
- **Preparación**: se pueden realizar cuando están asignados los RRHH y una versión, y deberían concluirse antes del comienzo de la versión objetivo en la cual se implementará la unidad de trabajo. Incluye el análisis, las revisiones y estimaciones.
- **Implementación**: se realizan durante la versión objetivo. Incluye la implementación, aplicación de pruebas e implantación.

Las actividades de cada workflow pueden variar significativamente dependiendo de factores tales como: cantidad y especialización de agentes participantes, validaciones o negociaciones predeterminadas con el cliente, características del producto (necesidad de migración, traducción, etc.), niveles y actividades de pruebas (unitarias, de integración, de aceptación, pruebas de regresión, automatización de pruebas), etc. Cada unidad de trabajo en una iteración del proyecto podría tener su propio workflow. Sin embargo, en la práctica basta con disponer de un reducido conjunto de workflows que permitan cubrir los tipos de unidades de trabajo que se presentan en el proyecto.

Los estados en los que se puede encontrar una actividad (asociada a una unidad de trabajo) son los siguientes:

- **Por Llegar**: el agente está asignado a la actividad pero aún no ha recibido la unidad de trabajo pues está en alguna actividad anterior en el workflow.
- **Pendiente**: el agente ha recibido la unidad de trabajo en la actividad pero aún no ha comenzado a trabajar en ella.
- **Activa**: el agente está trabajando en la actividad (y el sistema está registrando tiempo dedicado a ella).
- **Pausada**: el agente ha interrumpido su trabajo en la actividad (y se ha detenido el registro de tiempo.
- **Finalizada**: el agente ha terminado la actividad (la unidad de trabajo ha pasado automáticamente a las actividades siguientes en el workflow asociado).
- **Omitida**: la actividad ha sido omitida, es decir, se ha decidido no realizar la actividad (se ha saltado hacia adelante en el workflow). A menos que la unidad de trabajo de un salto hacia atrás, la unidad de trabajo no pasaría por la actividad.

3 Gestión del tiempo en TUNE-UP

En cualquier tipo de proyecto a priori es interesante contar con una previsión del ordenamiento de las actividades, de los tiempos máximos y mínimos de comienzo y finalización de las actividades, y de las holguras asociadas. Sin embargo, cuando se trata de un proyecto de desarrollo de software, establecer esto puede ser muy complejo y poco efectivo, especialmente si se pretende que los agentes respondan de forma ágil y autónoma ante los cambios cuando los plazos de entrega son reducidos. Así, en un proyecto de software son más protagonistas aspectos como los siguientes:

- Los agentes en general no realizan una actividad de forma ininterrumpida. Frecuentemente es necesario discutir alternativas o analizar elementos previamente no considerados, o simplemente hay cambios en las prioridades. Esto provoca que los agentes detengan actividades y continúen con otras, para posteriormente retomar aquellas detenidas. Además, es inevitable el retrabajo generado por saltos atrás en el proceso, como por ejemplo desde actividades de pruebas hacia actividades de implementación, cuando se detectan defectos.
- Para el ordenamiento y restricciones de tiempo entre actividades deben analizarse todas las actividades del proyecto (o incluso de otros proyectos si los recursos son compartidos) pues el trabajo en general es colaborativo y el retraso o anticipación en el trabajo de un agente tiene complejas consecuencias en los tiempos.
- Cada iteración o versión del producto suele incluir muchas unidades de trabajo, cada una de ellas siguiendo un workflow específico y siendo realizada por agentes asignados. Un grafo que represente la precedencia de actividades puede llegar a tener fácilmente cientos o incluso miles de nodos.
- Los workflows tienen una expresividad mucho mayor que la abordada en diagramas de red PERT/CPM (los métodos más populares utilizados para representar y analizar los tiempos en un proyecto). Así, dichos métodos deben ser adaptados para poder aplicarse a workflows [9].

Por lo anterior, en TUNE-UP, como una primera aproximación a dichos cálculos de tiempo, trabajamos con lo que denominamos “holgura simple” del agente, la cual sólo considera el tiempo restante con respecto del tiempo disponible. Con la

información que se recolecta en TUNE-UP es muy sencillo y rápido calcular dicha holgura simple. Así, cuando la holgura simple de algún agente es negativa (o positiva pero muy pequeña) podemos asegurar que el proyecto está desnivelado. Sin embargo, cuando el agente tiene teóricamente suficiente disponibilidad respecto de las horas restantes, no se puede asegurar que el agente se encuentre en situación de desbalanceo de carga puesto que no estamos considerando que el trabajo depende también de otros agentes. Aún así, la holgura simple nos ha resultado útil como indicador de sobrecarga del agente y síntoma de riesgo en los compromisos de una versión. La gestión del tiempo en TUNE-UP incorpora fundamentalmente las siguientes prácticas:

- Dividir el trabajo en unidades de trabajo que sean convenientes no sólo de cara a la captura y negociación de requisitos con el cliente sino también para la planificación y seguimiento del proyecto. De forma orientativa una actividad no debería suponer más de una semana de trabajo para un agente. Esto facilita el seguimiento continuo del trabajo.
- Definición del workflow de cada unidad de trabajo previendo el flujo de actividades por las cuales pasará.
- Asignación de unidades de trabajo a agentes considerando el balanceo de la carga de trabajo de los agentes.
- Estimación (y re-estimación) del esfuerzo para realizar las actividades asociadas a la realización de las unidades de trabajo.
- Registro actualizado del tiempo invertido en actividades. Cada vez que un agente realiza una actividad (y la finaliza) se genera un registro de seguimiento de tiempo, con lo cual, es sencillo distinguir entre el trabajo realizado la primera vez y el posterior retrabajo.
- Seguimiento continuo del estado del proyecto posibilitando las acciones correctivas oportunas.

Tabla 1. Abreviaturas usadas en las formulas de tiempo.

Abreviatura	Significado
T_E	Tiempo estimado
T_R	Tiempo registrado
T_P	Tiempo del período, considerado como una fracción respecto al tiempo en un mes
TA_M	Tiempo contractual del agente en un mes
TN_P	Tiempo no considerado en el período, por festivos, ausencias, permisos, bajas, etc.
TE_P	Tiempo extra del agente en el período
TR_P	Tiempo restante para finalizar actividades en el período
TD_P	Tiempo disponible para abordar actividades en el período
TH_P	Tiempo de holgura simple del agente en el período

$$TR_P = \sum_{Actividades}(T_E - T_R) \quad (1)$$

$$TD_P = T_P * TA_M - TN_P + TE_P \quad (2)$$

$$TH_P = TD_P - TR_P \quad (3)$$

La fórmula (1) permite obtener el tiempo restante del agente sumando las diferencias entre tiempo estimado y real para cada actividad asignada y que deba

entregarse en el período analizado. La fórmula (2) determina el tiempo disponible del agente en un período a partir del día de hoy. Con el resultado de (1) y (2) conseguimos la holgura simple del agente reflejada en (3).

Así, por ejemplo si quisiéramos conocer la holgura simple de un agente en las próximas dos semanas (T_P). Calcularíamos la suma de las diferencias entre estimaciones y registros de tiempo para todas sus actividades que tienen fecha de entrega en las próximas dos semanas, supongamos que esta suma nos da un tiempo restante de 65 hrs (TR_P). Suponiendo que el agente trabaja 120 hrs/mes (TA_M), y que no existe tiempo a descartar en el período ($TN_P = 0$) ni tiempo extra ($TE_P = 0$), entonces el tiempo disponible para el agente en el período son 60 hrs (TD_P= 0.5*120 – 0 + 0). Así, finalmente la holgura simple para el agente es – 5 hrs, es decir, a día de hoy el agente no podría cumplir con sus compromisos.

La implementación de dichos cálculos de holgura es más compleja pues debe considerar otros elementos (ya incorporados en TUNE-UP Process Tool) como por ejemplo:

- Si un agente trabaja en varios productos a la vez, puede interesar fijar porcentajes de distribución de tiempo del agente en cada producto. Así, a nivel global el agente puede tener una cierta situación de holgura y ésta puede ser diferente cuando se analiza a nivel de actividades de un determinado producto. De forma similar puede ser interesante que el agente tenga porcentajes prefijados de distribución de tiempo respecto de las actividades, es decir, su situación de holgura puede ser incluso relativa a una determinada actividad.
- Cuando el tiempo registrado sobrepasa al tiempo estimado el cálculo de tiempo restante se invalida pues se considerarían tiempo restante negativo para una actividad. En estos casos se descarta dicho tiempo y se alerta de la situación para que el agente proceda a una re-estimación.
- La flexibilidad del workflow de la unidad de trabajo obliga a hacer reajustes de estado de las actividades cuando se produce re-asignación del agente de una actividad, saltos hacia atrás o hacia adelante en el workflow o incluso cambio del workflow de una unidad de trabajo.
- Todas las actividades no finalizadas y con fecha de entrega anterior al período consultado deben ser consideradas en dicho período como actividades retrasadas.

4 TUNE-UP Process Tool

La mejor forma de ilustrar la gestión de tiempos en TUNE-UP es describir cómo a este aspecto se le ha dado apoyo mediante la herramienta TUNE-UP Process Tool. Un aspecto clave para el proyecto es que los agentes se dediquen a las actividades acertadas de acuerdo con las prioridades. El agente decide qué actividad realizar dependiendo de los plazos de la versión (fechas de inicio y de término), la prioridad de la unidad de trabajo en la versión, y la posición de la actividad en el workflow. El Planificador Personal (Fig. 2) ayuda a los agentes a conocer las unidades de trabajo que tiene asignadas y el estado en el que se encuentran. Cada unidad está en una o más actividades en un instante de tiempo (si su workflow permite el paralelismo de dichas actividades) y asignada a un único agente por actividad. El grid de la derecha

en la Fig. 2 muestra información de cada actividad incluyendo: producto, versión, descripción de la unidad de trabajo, tiempos calculados, estado actual de la actividad, etc. El grid de la izquierda de la Fig. 2 resume las unidades de trabajo según la actividad y el estado en el cual se encuentran. Con los tiempos calculados el agente puede visualizar (ver columnas del grid de la derecha de la Fig. 2) y así conocer el tiempo que lleva registrado en cada actividad (T.Real), los tiempos que ha estimado (T. Est., estimado y estimado ajustado), el porcentaje completado con respecto el estimado (%Com.) y las horas restantes (H. Rest.) que le quedan para finalizar la actividad.

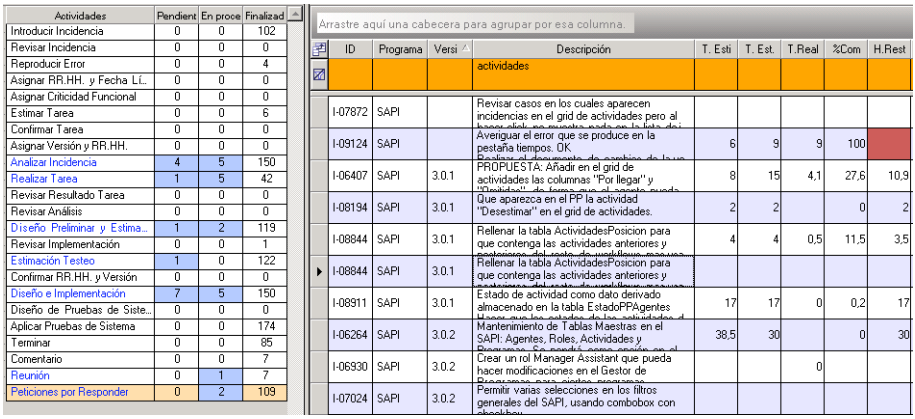


Fig. 2. Fragmento de interfaz del Planificador Personal

Cuando el tiempo registrado supera al que se ha estimado se alerta coloreando la celda H. Rest. Dicho tiempo (que sería negativo) se considera como valor nulo en el cómputo. El agente debería re-estimar la actividad en esta situación cuanto antes.

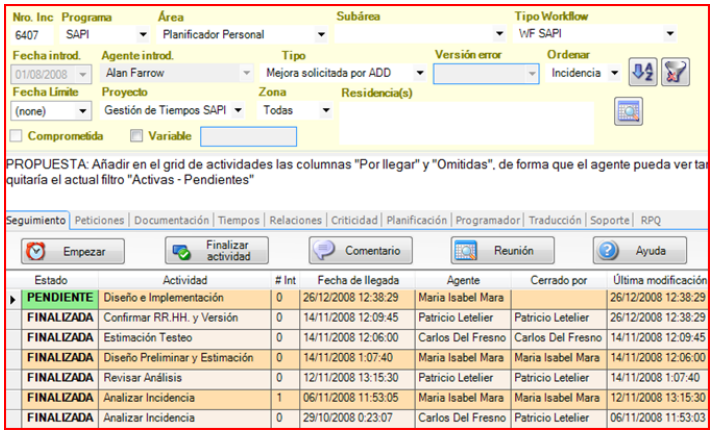
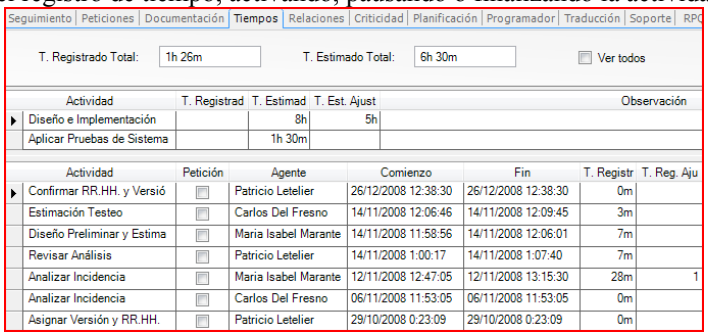


Fig. 3. Fragmento de interfaz del Gestor de Unidades de Trabajo

Cuando el agente decide en qué actividad trabajar, accede al Gestor de Unidades de Trabajo (Fig. 3) el cual le apoya en la realización de las actividades de una unidad de trabajo. En la parte superior de la Fig.3 se observan los datos generales de la unidad de trabajo. En la parte inferior se observa la pestaña Seguimiento la cual resume los

registros de seguimiento, es decir, una línea generada cada vez que se pasa a una actividad en el workflow de la unidad de trabajo (incluyendo fechas-horas de inicio y finalización, agente, y otros datos). El Gestor de Unidades de Trabajo ofrece además otras pestañas entre las que destacan: Peticiones (mecanismo de comunicación entre los agentes en el contexto de una unidad de trabajo), Documentación (espacio de documentación para compartir especificaciones de la unidad de trabajo), Relaciones (relaciones de dependencia de la unidad de trabajo con respecto de otras), Planificación (asignación de agentes a actividades de la unidad de trabajo y asignación a una versión del producto) y Tiempos, que explicaremos a continuación.

Con los botones Comenzar/Continuar/Pausar (es el mismo botón que cambia de nombre según el estado de la actividad) y Finalizar Actividad el agente puede controlar el registro de tiempo, activando, pausando o finalizando la actividad.


Seguimiento | Peticiones | Documentación | Tiempos | Relaciones | Criticidad | Planificación | Programador | Traducción | Soporte | RPC

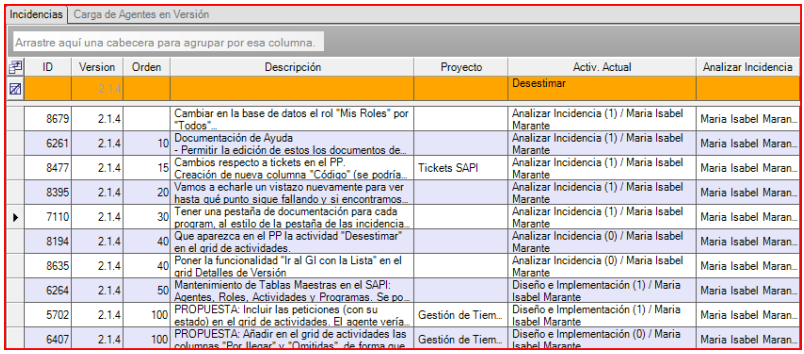
T. Registrado Total: 1h 26m T. Estimado Total: 6h 30m Ver todos

Actividad	T. Registrad	T. Estimad	T. Est. Ajust	Observación
Diseño e Implementación		8h	5h	
Aplicar Pruebas de Sistema		1h 30m		

Actividad	Petición	Agente	Comienzo	Fin	T. Registr	T. Reg. Aju
Confirmar RR.HH. y Versió		Patricio Letelier	26/12/2008 12:38:30	26/12/2008 12:38:30	0m	
Estimación Testeo		Carlos Del Fresno	14/11/2008 12:06:46	14/11/2008 12:09:45	3m	
Diseño Preliminar y Estima		Maria Isabel Marante	14/11/2008 11:58:56	14/11/2008 12:06:01	7m	
Revisar Análisis		Patricio Letelier	14/11/2008 1:00:17	14/11/2008 1:07:40	7m	
Analizar Incidencia		Maria Isabel Marante	12/11/2008 12:47:05	12/11/2008 13:15:30	28m	1
Analizar Incidencia		Carlos Del Fresno	06/11/2008 11:53:05	06/11/2008 11:53:05	0m	
Asignar Versión y RR.HH.		Patricio Letelier	29/10/2008 0:23:09	29/10/2008 0:23:09	0m	

Fig. 4. Fragmento de Pestaña Tiempos

La pestaña Tiempos (Fig. 4) ofrece funcionalidad específica de tiempos para la unidad de trabajo. En la parte superior los agentes establecen las estimaciones para cada actividad. En la parte inferior se muestran los registros de tiempo producidos automáticamente por los pares de acciones Comenzar/Continuar – Pausar/Finalizar. Tanto en las estimaciones como en los registros de tiempo el agente puede realizar ajustes cuando por ejemplo, prevea que la estimación anterior no se va a cumplir, o cuando haya olvidado Activar, Pausar o Finalizar.


Incidencias | Carga de Agentes en Versión

Arrastre aquí una cabecera para agrupar por esa columna.

ID	Version	Orden	Descripción	Proyecto	Activ. Actual	Analizar Incidencia
	2.1.4				Desestimar	
8679	2.1.4		Cambiar en la base de datos el rol "Mis Roles" por "Todos"...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
6261	2.1.4	10	Documentación de Ayuda - Permitir la edición de estos los documentos de...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
8477	2.1.4	15	Cambios respecto a tickets en el PP. Creación de nueva columna "Código" (se podría...	Tickets SAPI	Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
8395	2.1.4	20	Vamos a echarle un vistazo nuevamente para ver hasta qué punto sigue fallando y si encontramos...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
7110	2.1.4	30	Tener una pestaña de documentación para cada program, al estilo de la pestaña de las incidencia...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
8194	2.1.4	40	Que aparezca en el PP la actividad "Desestimar" en el grid de actividades.		Analizar Incidencia (0) / Maria Isabel Marante	Maria Isabel Maran...
8635	2.1.4	40	Poner la funcionalidad "Ir al GI con la Lista" en el grid Detalles de Versión		Analizar Incidencia (0) / Maria Isabel Marante	Maria Isabel Maran...
6264	2.1.4	50	Mantenimiento de Tablas Maestras en el SAPI: Agentes, Roles, Actividades y Programas. Se po...		Diseño e Implementación (1) / Maria Isabel Marante	Maria Isabel Maran...
5702	2.1.4	100	PROPUESTA: Incluir las peticiones (con su estado) en el grid de actividades. El agente vería...	Gestión de Tiem...	Diseño e Implementación (1) / Maria Isabel Marante	Maria Isabel Maran...
6407	2.1.4	100	PROPUESTA: Añadir en el grid de actividades las columnas "Por llegar" y "Omitidas", de forma que...	Gestión de Tiem...	Diseño e Implementación (0) / Maria Isabel Marante	Maria Isabel Maran...

Fig. 5. Fragmento de interfaz Gestión de Productos – Unidades de Trabajo de la Versión

Para realizar la planificación y seguimiento el Product Manager utiliza el módulo Gestión de Productos (donde se gestionan las versiones de los productos) y desde allí, seleccionando un producto y versión se accede a Unidades de Trabajo en la Versión

(Fig.5) y a la Carga de Agentes en la Versión (Fig.6). Así, en la interfaz mostrada en la Fig.5, el Product Manager puede consultar los datos resumidos de cada unidad de trabajo en la versión (en particular, puede conocer la actividad actual y el agente asignado a cada unidad de trabajo) o acceder al Gestor de Unidades de Trabajo con una determinada. También puede establecer el orden de prioridad de las unidades de trabajo en una versión o cambiar su versión.

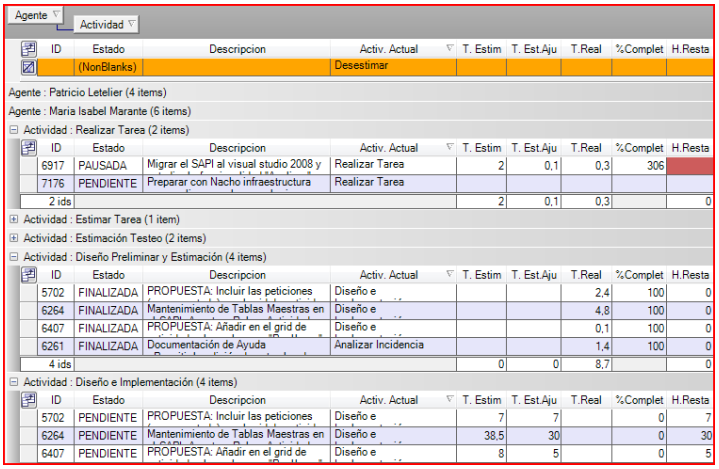


Fig. 6. Fragmento de interfaz Gestión de Productos – Carga de Agentes en Versión

En la Fig. 6 se muestra la Pestaña Carga de Agentes en Versión. En ella se puede conocer en cualquier momento la holgura simple de los agentes respecto de sus actividades en una versión del producto. En esta interfaz se ofrecen potentes mecanismos de filtros y agrupaciones por columnas. Cuando una versión tiene problemas de holgura, en esta misma interfaz el Product Manager puede cambiar el agente asignado a la actividad (para balancear la carga de un agente) o cambiar de versión alguna unidad de trabajo. Otras alternativas son modificar la fecha de término de la versión, asignar más recursos humanos a la versión o dividir unidades de trabajo para realizarlas incrementalmente en varias versiones.

5 Trabajos relacionados

Modelos de referencia como CMMI y estándares como ISO 90003 y SPICE, así como otras guías para gestión de proyectos tales como el PMBOK (Project Management Book of Knowledge) [13] sólo recomiendan prácticas muy generales para la gestión del tiempo. Así, corresponde a las metodologías el definir mecanismos concretos para aplicar dichas prácticas. Las metodologías ágiles aciertan respecto de la granularidad en la definición de las unidades de trabajo, su asignación y estimación, y en el seguimiento continuo. Además, promueven en cierto grado el registro actualizado de tiempos invertidos. Sin embargo, su definición de roles tan genéricos y la no existencia de workflows explícitos, resulta ser incompatible con enfoques de desarrollo basados en mayor especialización de roles junto con mayores necesidades

de coordinación. Por otra parte, cuando se utiliza una metodología tradicional en los planes se suele asumir un modelo de proceso en cascada, lo cual resulta fácil de elaborar y entender (algo positivo para cerrar un acuerdo con el cliente), pero contraproducente para el control y seguimiento si dicho modelo de proceso no es el más apropiado para el proyecto. Aunque las metodologías tradicionales ponen más énfasis en el trabajo colaborativo basado en workflows, no ofrecen mecanismos para su aplicación. Tampoco promueven la asignación ni estimación detallada respecto de esfuerzos estimados o registrados por los agentes.

Existe una gran cantidad de herramientas genéricas para gestión de proyectos, sin embargo, éstas presentan inconvenientes importantes para la gestión de tiempos en proyectos software. Por una parte, no soportan adecuadamente un proceso iterativo e incremental basado en unidades de trabajo que se implementan de forma colaborativa. Por otra parte, no promueven/soportan un registro actualizado de estimaciones y tiempos invertidos pues en la práctica se usan desconectadas del trabajo de los agentes, los cuales periódicamente deberían actualizar los datos del proyecto. En los últimos años han aparecido diversas herramientas para gestión ágil de proyectos software, las cuales sí que asumen que el proceso de desarrollo es iterativo e incremental y basado en unidades de trabajo. Herramientas tales como Rally (www.rallydev.com), TargetProcess (www.targetprocess.com), ScrumWorks (www.danube.com) y VersionOne (www.versionone.com) presentan un especial atractivo respecto de su interfaz y plataforma Web. Sin embargo, en cuanto a gestión de tiempos los mecanismos ofrecidos son muy rudimentarios y ninguna de ellas utiliza workflows para la gestión de unidades de trabajo.

Los trabajos de J. Eder y E. Panagos en el área de sistemas de workflows [4, 5] ilustran el interés en mecanismos para la gestión del tiempo y especificación de restricciones de tiempo. Estos trabajos, si bien no están enmarcados en workflows para procesos de desarrollo de software, resultan complementarios a nuestra propuesta. De hecho, la recolección y cómputo de tiempo ofrecidos por TUNE-UP constituirían la base para poder aplicar dichos mecanismos.

Los llamados Process-Centered Software Engineering Environments [6] impulsaron gran actividad de investigación a fines de los 90's, sin embargo, hasta donde sabemos, dicha línea de investigación desapareció y sus resultados no llegaron a aplicarse en el ámbito industrial. Los trabajos de M. Richter, D. Rombach y F. Maurer respecto del entorno MILOS [14, 15] son representativos de dicha generación de herramientas. De estos trabajos pueden extraerse algunas ideas para una gestión más sofisticada de los tiempos y del soporte para el trabajo de los agentes, por ejemplo, gestión integrada de la planificación y de los workflows, almacenamiento y recuperación de experiencias para ser reutilizadas, ambas en [15], o la combinación de enfoques workflows y groupware para flexibilizar la aplicación de técnicas de workflows en contextos de procesos menos estructurados [3]. No descartamos, que un futuro, evaluemos la incorporación en TUNE-UP de algunas de esas ideas.

6 Conclusiones

La adecuada gestión de los tiempos en un proyecto permite tomar acciones correctivas oportunas. Tanto el Product Manager como cada uno de los agentes pueden detectar desajustes en el cumplimiento de compromisos observando las holguras. Sin embargo, estos beneficios sólo se alcanzan cuando se dispone de la información actualizada y completa de los tiempos del proyecto. Sin embargo, en cuanto a precisión ningún mecanismo es perfecto, pues en las estimaciones de tiempos y de disponibilidades de agentes intervienen muchos factores no predecibles ni controlables totalmente, por ejemplo: cambios en los requisitos, modificación de prioridades, imprevistos que afectan la disponibilidad del agente, etc.

Es reconocido que la implantación de una metodología es un proceso largo y que presenta dificultades, tales como la formación y entrenamiento del equipo o la resistencia al cambio por parte de algunos agentes. Una gestión del tiempo como la ofrecida por TUNE-UP conlleva además otros obstáculos:

- Disciplina de estimación y registro de tiempo. Hay que ofrecer pautas para realizar la estimación y entrenar a los agentes en ello. Las acciones asociadas al registro de tiempo deben ser lo menos molestas para los agentes.
- Desconfianza de los agentes al posible control basado en los tiempos. Es difícil convencer a los agentes de que las ventajas de esta gestión de tiempos van mucho más allá de su posible uso para su control. Sin embargo, a la larga el agente termina valorando la información de sus registros de tiempo y el análisis que puede hacerse de ella. La trasparencia contribuye a ganarse la confianza y aceptación de los agentes. Para ello es fundamental proporcionar a cada agente toda la información de tiempos que de él recolecta el sistema.
- Evitar confundir la gestión de tiempos con el reloj para fichar la entrada y salida de la empresa. El sistema no pretende que el tiempo registrado mensualmente sea igual al tiempo contractual, aunque podría llegar a serlo sería demasiado molesto tener que registrar tiempos para cada tipo de tarea posible de desarrollar. Por esto es importante acotar las tareas para las cuales es relevante que el agente registre tiempos.
- Introducir la disciplina de planificación y gestión de tiempos en la rutina de trabajo de los agentes. Esto exige que el agente tenga en todo momento en ejecución TUNE-UP Process Tool, lo cual en un principio no era valorado. Sin embargo, a medida que la herramienta fue incorporando funcionalidad respecto a planificación y seguimiento del trabajo, comunicación y coordinación entre agentes, espacio compartido de documentos, etc., ésta se ha transformado en un apoyo imprescindible para el trabajo de los agentes. Un elemento muy importante respecto de la comodidad para los agentes fue la instalación de dos monitores en cada puesto de trabajo. Esto les permite tener las ventanas de la herramienta en un monitor y utilizar el otro para el trabajo específico según sus roles.

TUNE-UP ha evolucionado en el contexto de una PYME de desarrollo de software bajo el marco de varios proyectos universidad-empresa. Comenzamos con la definición e implantación de un proceso dirigido por las pruebas de aceptación y al mismo tiempo fuimos desarrollando y utilizando la herramienta de apoyo TUNE-UP Process Tool. Esta herramienta con sus principales funcionalidades para gestión de

tiempos lleva 2 años en funcionamiento y satisface las necesidades de desarrollo y mantenimiento de un producto de tamaño considerable (un ERP dirigido al sector socio-sanitario) y de cinco productos más pequeños, extensiones de dicho ERP. En el ERP trabaja un equipo con 3 analistas, 4 testers y 6 programadores, con versiones de alrededor de un mes que incluyen entre 50 y 100 solicitudes de cambios. TUNE-UP Process Tool en sí misma es un producto de la empresa, que sigue la metodología TUNE-UP y utiliza la herramienta. En el desarrollo de TUNE-UP Process Tool participan 2 programadores. Actualmente tenemos definidos 20 workflows, algunos específicos para un producto, otros compartidos entre diferentes productos, unos con pocas actividades, y otros con hasta 30 actividades. Cada producto utiliza alrededor de cinco workflows.

Referencias

1. Beck K. Extreme Programming Explained: Embrace Change. Addison-Wesley. 2000.
2. CMMI v1.2. Capability Maturity Model Integrated. SEI–Software Engineering Institute, Carnegie Mellon. 2009. www.sei.cmu.edu/cmmi/models/ (visitado junio de 2009).
3. Dellen B., Maurer F., Pews G. Knowledge Based Techniques to Increase the Flexibility of Workflow Management. Data & Knowledge Engineering 29, 1997. Elsevier.
4. Eder J., Panagos E. Managing Time in Workflow Systems. Workflow handbook 28, 109-132. 2001.
5. Eder J., Panagos E., Rabinovich M. Time Constraints in Workflow Systems. Lecture Notes in Computer Science 15, 165-280. Springer. 1999.
6. Gruhn V. Process-Centered Software Engineering En.vironments: A Brief History and Future Challenges. Annals of Software Engineering 14, 363-382. Kluwer Academic Publishers. 2002.
7. Hiranabe K. Kanban Applied to Software Development: from Agile to Lean. www.infoq.com/articles/hiranabe-lean-agile-kanban (visitado junio de 2009)
8. Hyun, J., Sun J., Ho, M. Extracting the workflow critical path from the extended well-formed workflow schema. Journal of Computer and System Sciences 70 (2005) 86–106. Academic Press, Inc.
9. ISO-International Organization for Standardization. ISO/IEC 15504, SPICE: Software Process Improvement and Capability dEtermination. 1998.
10. ISO-International Organization for Standardization. ISO/IEC 90003, Software engineering-Guidelines for the application of ISO 9001:2000 to computer software. 2004.
11. Kroll P., Kruchten P. Booch G. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley Professional. 2003.
12. MAP-Ministerio de Administraciones Públicas de España. Métrica Versión 3, Metodología de planificación, desarrollo y mantenimiento de sistemas de información. 2005. www.csi.map.es/csi/metrica3/ (visitado en junio de 2009)
13. PMI-Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide) - Fourth Edition. 2008.
14. Richter M., Maurer F. MILOS and MASE: Past & Present. ase.cpsc.ucalgary.ca /uploads/Publications/Milos_22.08.2003.pdf (visitado en junio de 2009)
15. Rombach D. A process Platform for Experience-Based Software Development. Proc. of Int. Colloquium of the Sonderforschungsbereich 501 U.of Kaiserslautern, pp.47-57. 2003
16. Schwaber K., Beedle M. Agile Software Development with Scrum. Prentice Hall, Series in Agile Software Development. 2001.
17. Watts, H. Introducción al proceso de software personal. Addison-Wesley Iberoamericana. 2001.

Selección de Herramientas para la Gestión de Proyectos de Software en Pequeñas y Medianas Empresas

Lornel Rivas[1], María Pérez[1], Luis E. Mendoza[1] y Anna Grimán[1]

[1]Departamento de Procesos y Sistemas, Universidad Simón Bolívar,
Apartado Postal 89000, Caracas 1080-A, Venezuela
{lrivas, movalles, lmendoza, agriman}@usb.ve

Abstract. La gestión de proyectos (GP) de software se apoya en herramientas software que soportan la planificación, el seguimiento y el control del proyecto. Por su parte, la selección de herramientas debe responder a las necesidades y características de la empresa. Esto es especialmente importante en las Pequeñas y Medianas Empresas (PyMES), las cuales toman decisiones bajo diversas restricciones. Proponemos en este trabajo criterios para la selección de herramientas software de apoyo a la GP en PyMES. Apoyándonos en el enfoque Objetivo-Pregunta-Métrica, proponemos la operacionalización de los criterios a través de 94 métricas. Éstas fueron aplicadas en cinco herramientas de soporte a la GP: Gantt Project, Trac, Enterprise Architect, DotProject y OpenProj. Los resultados han sido coherentes con el alcance de cada herramienta y nos han revelado sus ventajas y debilidades; no existiendo una de ellas que satisfaga completamente todos los objetivos de evaluación. Enterprise Architect destaca durante el Inicio y la definición del alcance, mientras que OpenProj, DotProject y Trac hacen lo propio en la Planificación, y el Seguimiento y control del proyecto. La evaluación realizada nos ha permitido validar la generalidad, viabilidad, coherencia, utilidad y repetibilidad de nuestra propuesta.

Keywords: Gestión de proyectos, Criterios de selección de herramientas, Calidad de software, Métricas, PyMES.

1 Introducción

Actualmente académicos y profesionales reconocen la influencia de la Gestión de Proyectos (GP) en el éxito de los proyectos software [1,2]. Aunque las razones que propician tal éxito son diversas, la GP destaca por sus contribuciones para alcanzar la entrega de un producto de calidad dentro del tiempo y presupuesto establecidos. Para lograr esto, la GP incorpora herramientas automatizadas de Ingeniería de Software (IS) que son clave en el apoyo al equipo de desarrollo en actividades como la planificación, el seguimiento y el control de los proyectos. En este sentido, el uso de las herramientas automatizadas ha sido clave en el éxito de tales actividades [1].

Esta situación resulta particularmente importante para las Pequeñas y Medianas Empresas (PyMES). La principal razón para ello es que las PyMES han incrementado su presencia en la industria del software a escala global; asumiendo proyectos cada vez más complejos y compartiendo restricciones que exigen ser eficientes en el

manejo de los recursos. Esto ha requerido también que las PyMES sean efectivas en aquellos procesos de evaluación y selección de herramientas software, para obtener así los beneficios esperados de éstas. No obstante, tomar la decisión sobre cuál herramienta software adquirir no es una tarea trivial; por el contrario, requiere analizar la empresa y sus necesidades. Su selección puede realizarse mediante diferentes aproximaciones: el uso de listas de comprobación [3]; la aplicación de métodos y metodologías de evaluación [4]; o el uso de estándares, tales como ISO/IEC 14102 o IEEE 1209 [5,6]. Sin embargo, existen dificultades para la aplicación de aquellos procesos definidos en los estándares de IS en el ámbito de las PyMES [7]. En [3] se justifica esta dificultad por el número y la complejidad de las características a evaluar, su organización interna, y la diversidad de aspectos organizacionales que pueden estar involucrados.

Por otra parte, aunque se han propuesto criterios de selección para ciertos dominios de la IS, no existen criterios específicos para aquellas áreas de especial interés para las PyMES; incluyendo la GP. Esto es analizado en [8], donde se abordan las dificultades en la aplicación de estándares de IS en áreas del ciclo de vida de desarrollo de software que incluyen explícitamente la GP. Por ello, persiste actualmente la necesidad de orientaciones más adecuadas que apoyen a las PyMES en aquellos procesos de evaluación y selección de herramientas software que apoyen la GP.

En etapas previas de esta investigación hemos propuesto criterios de carácter general dirigidos a la selección de herramientas en PyMES [9]. En el presente trabajo analizamos las actividades claves de la GP en PyMES y, con base en este análisis, proponemos un conjunto de criterios que apoyen a los decisores en la selección de herramientas adecuadas a sus necesidades.

En la propuesta de tales criterios hemos tomado en cuenta dos aspectos metodológicos fundamentales: (1) el contexto organizacional [3,4], y (2) su objetividad. Para satisfacer el primero de estos aspectos, hemos incorporado el análisis de aquellas características particulares de las PyMES. En relación al segundo aspecto, hemos utilizado el enfoque Objetivo-Pregunta-Métrica (GQM, por sus siglas en inglés) [10]. Una vez que hemos contado con un conjunto de criterios adecuados y objetivos, los hemos aplicado a cinco herramientas software (Gantt Project, Trac, Enterprise Architect, DotProject y OpenProj) para comprobar las bondades de nuestra propuesta.

Los resultados generales revelaron las ventajas de cada una de estas herramientas en estadios particulares del proyecto software; al tiempo que también se hicieron evidentes las fortalezas y debilidades comunes a todas ellas respecto a los objetivos propuestos por las PyMES. Esto demuestra no sólo la viabilidad de aplicar los criterios propuestos sino también la utilidad de éstos dentro del proceso de evaluación y selección.

En las próximas secciones establecemos un marco de referencia acerca de la GP en PyMES (sección 2). Sobre esta base, proponemos los criterios de selección expresados en un conjunto de 94 métricas (sección 3), las cuales aplicamos a las herramientas antes mencionadas (sección 4). Por último, presentamos las conclusiones obtenidas y los trabajos futuros (sección 5).

2 Marco para el Estudio de la Gestión de Proyectos en PyMES

Existen en el mercado numerosas herramientas software que apoyan actividades de GP. Sin embargo, su selección efectiva requiere del establecimiento de criterios basados en el contexto de la organización [4]. Ello implica el reconocimiento de las características del proyecto, así como de la empresa que hace la selección.

Respecto a los proyectos software, éstos comparten con cualquier otro tipo de proyecto, características y principios bien establecidos. Un proyecto software es definido entonces a partir de actividades precisas, tiempos delimitados y objetivos claros, que se orientan hacia el logro de un producto [11]. La GP surge ante la necesidad de coordinar tales actividades de forma sistemática y disciplinada. Para ello, el proyecto se organiza en las siguientes áreas [12]: (1) Inicio y definición del alcance, (2) Planificación, (3) Revisión y evaluación, (4) Cierre, y (5) Medición. Cuando hablamos de proyectos software, diversos factores pueden afectar la GP, entre los que destacan [13]: (1) el cambio tecnológico, (2) las necesidades de personal capacitado, (3) la intensiva relación con el usuario, y (4) el impacto de los cambios en el alcance. De allí la importancia de considerar las actividades de GP en la IS.

Por otra parte, las PyMES se caracterizan por ciertas restricciones que afectan sus procesos de adopción tecnológica, incluyendo los de evaluación y selección de tecnologías. Restricciones de recursos para la adquisición de tecnologías, la contratación de personal especializado, y el desarrollo de infraestructura —entre otras— agregan complejidad a la toma de decisiones en PyMES dedicadas al desarrollo de software. Sin embargo, las PyMES se caracterizan asimismo por un alto grado de flexibilidad, por lo que suelen ajustar sus procesos según las necesidades del proyecto y así generar respuestas ante las exigencias de productividad y calidad a las que son sometidas. Por ello, es común la adopción de enfoques simplificados para la gestión de sus proyectos [2]; requiriendo herramientas software que les proporcionen soporte, especialmente en aquellas tareas que les son prioritarias.

Aunque la GP es un tema ampliamente desarrollado en la literatura, la información sobre GP de software en el contexto de las PyMES es más escasa. No obstante, algunos autores como [14, 15, 16] reconocen la GP como una necesidad que comparten las PyMES que se dedican al desarrollo de software (ver Tabla 1). Por otra parte, autores como [2, 9, 8, 17] revelan coincidencias en la importancia que le otorgan a algunas tareas de GP específicas en PyMES (véase Tabla 2).

Tabla 1: Algunas necesidades de apoyo en PyMES desarrolladoras de software [14, 15, 16]

Necesidades de apoyo	Argumentación
Manejo de riesgos	Vulnerabilidad de las finanzas; necesidad de eficiencia
Estimación de tareas	Carencias de históricos y de grandes bancos de proyectos
Herramientas adecuadas	Debilidades en la experiencia Necesidad de facilitar procesos de adopción tecnológica
Organización y coordinación de roles	Flexibilidad en la asignación de roles Informalidad en reglas y procedimientos
Flexibilidad en la GP	Procesos ajustados a necesidades del proyecto
Determinación de requisitos	Informalidad de la comunicación, las reglas y procedimientos Comunicación intensiva con los clientes
Mecanismos para la comunicación	Incremento en prácticas de trabajo distribuidas Informalidad de infraestructura

Tabla 2: Coincidencias observadas sobre actividades claves para GP en PyMES [2, 9, 8 17]

Temas prioritarios	Laporte et al.	Murphy y Ledwith	Rivas et al.	O'Connor et al.
Definición del alcance		√		√
Estimación de tareas	√	√	√	√
Gestión de riesgos	√	√	√	
Seguimiento de proyectos	√	√	√	√
Control de Proyectos		√		√
Estimación de costos			√	√
Manejo de Agenda		√	√	√

Las características de las PyMES generan inquietudes en torno al soporte que pueden obtener de las herramientas para la GP. De acuerdo con [12] todas las áreas de conocimiento definidas en PMBOK son directamente relevantes a la GP de software, sin embargo en SWEBOK se abordan aquellos tópicos específicos de la IS. De allí que ante la diversidad de enfoques existentes para estudiar la GP y el amplio reconocimiento de ambos cuerpos de conocimiento consideramos las áreas y actividades definidas en SWEBOK [12] como punto de partida para identificar aquellas de mayor interés para estas empresas.

Sobre esta base, identificamos las siguientes áreas de soporte de las herramientas software: (1) las herramientas de *planificación y seguimiento* y las de *gestión de riesgos* soportan actividades de planificación y ejecución del proyecto al facilitar la medición de esfuerzo, la estimación de costos, la programación de agendas, así como la identificación, la estimación y el seguimiento de riesgos [12], (2) las herramientas *de planeación*, apoyan la planificación mediante la generación de diagramas PERT, estimaciones diversas y hojas de cálculo [18], (3) las herramientas de *requisitos* apoyan actividades de especificación y gestión de requisitos [12]. En la Tabla 3 relacionamos las áreas y actividades de GP que consideramos revisten especial relevancia para las PyMES, y las herramientas software mencionadas anteriormente.

Tabla 3: Actividades de GP y herramientas de IS

Áreas de la GP	Actividades	Herramientas software
Inicio y definición del alcance del proyecto	Determinar y negociar requisitos Análisis de factibilidad Revisión de requisitos	Herramientas de requisitos de software Herramientas de planificación y seguimiento Herramientas de gestión de riesgos Herramientas de planeación
Planificación del Proyecto	Planificación Determinar Entregables Estimación de Esfuerzo, Agenda y Estimaciones de Costos Asignación de Recursos Gestión de Riesgos	
Ejecución del proyecto	Seguimiento y Control	

En este punto, es importante mencionar que existen áreas de conocimiento definidas en el PMBOK tales como la Gestión de Comunicaciones, la Gestión de Recursos Humanos, la Gestión de Integración, la Gestión de Calidad y la Gestión de Procura, que no forman parte del alcance de este artículo. Ello obedece a que diversas actividades que las conforman no se justifican de manera determinante en el ámbito de las Pymes, bien sea por razones como el reducido número de personal (lo que

afecta la Gestión de Recursos Humanos y la Gestión de Integración, por ejemplo), así como otras características que las definen, como la flexibilidad en su comunicación interna y externa, o la sencillez de sus estructuras organizativas (ambas afectan a su vez la Gestión de Comunicaciones). Sin embargo, estas áreas serán también analizadas en futuras investigaciones.

3 Propuesta de Criterios para la Selección de Herramientas Software para la GP en PyMES

El análisis previo nos permitió identificar ciertas actividades de GP que requieren un soporte automatizado. De acuerdo con los estándares ISO/IEC 14102 e IEEE 1209, los criterios de selección de herramientas software deben ser precisos y susceptibles de ser medidos [3, 4]. Por ello, hemos utilizado el enfoque GQM [5] para la definición de nuestros criterios, de la siguiente manera: (1) Un nivel conceptual: se establece un conjunto de *objetivos* orientadores del ejercicio de evaluación y selección; (2) Un nivel operativo: se propone un conjunto de *preguntas* que buscan el logro de tales objetivos; (3) Un nivel cuantitativo: se formulan las métricas que nos permiten capturar datos asociados a cada una de tales preguntas y así conocer la presencia o no de funcionalidades que debe tener una herramienta de GP.

La Tabla 4 resume los objetivos y preguntas propuestas para los niveles conceptual y operacional, respectivamente; así como el número de métricas correspondiente a cada pregunta (nivel cuantitativo).

Tabla 4: Objetivos, preguntas y número de métricas formuladas

Objetivos	Preguntas formuladas y número de métricas correspondientes	Núm. de métricas
O1.Conocer el soporte de la herramienta durante el Inicio y Definición del alcance	P1: ¿Soporta actividades para la definición de requisitos?	12
	P2: ¿Soporta actividades para el análisis de factibilidad?	5
	P3: ¿Soporta actividades para la revisión de requisitos?	3
O2. Conocer el soporte de la herramienta durante la Planificación	P1: ¿Soporta actividades para la planificación?	9
	P2: ¿Soporta actividades para la definición de entregables?	6
	P3: ¿Soporta actividades para la estimación de esfuerzos y costos?	27
	P4: ¿Soporta actividades para la asignación de recursos?	8
	P5: ¿Soporta actividades para la gestión de riesgos?	8
O3. Conocer el soporte de la herramienta en la Ejecución	P6: ¿Soporta actividades para el seguimiento y control?	16

Como se observa en la Tabla 4, hemos propuesto un total de 94 métricas que nos permiten dar respuesta a seis preguntas y tres objetivos de evaluación. Por razones de espacio sólo presentamos algunos ejemplos de estas métricas (ver Tabla 5). Nótese que las métricas propuestas representan aquellas funcionalidades esperadas de la herramienta software, y son medidas según la escala propuesta para cada una de ellas, con valores normalizados de 1 a 5.

Tabla 5: Ejemplos de métricas para los objetivos planteados

Objetivo 1: Conocer el soporte de la herramienta durante el Inicio y definición del alcance	
Pregunta 1: ¿Soporta actividades para la definición de requisitos?	
Permite asignar estados a los requisitos: a) abierto, b) en revisión, c) en ejecución, c) cerrado	[5-Todos los mencionados, 4- tres de los mencionados , 3 –dos de los mencionados – 2 uno de los mencionados –1 ninguno de los mencionados]
Permite crear nuevos tipos de requisitos	[5-si, 1- no]
Permite registrar fecha de inicio y fin del proyecto	[5-si, 1- no]
Permite registrar porcentaje de avance del proyecto	[5-si, 1- no]
Recupera elementos de datos históricos de otros proyectos	[5- Costos y tiempos, 4- sólo costos, 3- sólo tiempos, 2- otros datos, 1- ninguno]
Objetivo 2: Conocer el soporte de la herramienta durante la Planificación	
Pregunta 1: ¿Soporta actividades para la planificación?	
Permite definir fases de desarrollo del proyecto	[5-si, 1-no]
Permite registrar fechas de inicio y fin para cada fase	[5-si, 1-no]
Permite definir entregables para cada fase	[5-si, 1-no]
Permite establecer tareas dirigidas a los entregables	[5-si, 1-no]
Permite registrar fechas de inicio y fin de la iteración	[5-si, 1-no]
Pregunta 3: ¿Soporta actividades para la estimación de esfuerzos y costos?	
Permite registrar fechas de inicio y finalización de tareas	[5-si, 1-no]
Permite registrar número de horas por tarea	[5-si, 1-no]
Presenta parámetros de costos: (a) hardware, (b) software y mantenimiento, (c) viajes y capacitación, (d) esfuerzo	[5-Todos los mencionados, 4- tres de los mencionados , 3- dos de los mencionados, 2- uno de los mencionados, 1- ninguno de los mencionados]
Permite crear diagramas Gantt que reflejen responsables de cada tarea	[5-si, 1-no]
Permite crear gráficos de redes que reflejen dependencia entre tareas	[5-si, 1-no]
Pregunta 5: ¿Soporta actividades para la gestión de riesgos?	
Permite registrar descripción de riesgos	[5 si, 1 no]
Permite establecer prioridades a los riesgos: a) muy alta, b) alta, c) media, d) baja	[5-Todos los mencionados, 4- tres de los mencionados , 3 –dos de los mencionados – 2 uno de los mencionados –1 ninguno de los mencionados]
Permite registrar nuevos tipos de riesgos	[5-si, 1- no]
Permite establecer respuestas a riesgos	[5-si, 1- no]
Permite generar vistas sobre riesgos y sus elementos	[5-si, 1- no]
Objetivo 3: Conocer el soporte de la herramienta en la Ejecución	
Pregunta 1: ¿Soporta actividades para el seguimiento y control?	
Genera recordatorios de tareas	[5-si, 1-no]
Permite registrar acciones correctivas sobre tareas: (a) acción a tomar, (b) asignación de responsable, (c) fechas de realización, (d) autor	[5-Todos los mencionados, 4- tres de los mencionados , 3 –dos de los mencionados – 2 uno de los mencionados –1 ninguno de los mencionados]
Permite registrar incidentes	[5-si, 1- no]
Presenta horas trabajadas vs. horas planificadas	[5-si, 1- no]
Presenta recursos ejecutados vs. recursos ejecutados	[5-si, 1-no]

Al basarnos en [12], los criterios incluyen un sub-conjunto de preguntas/métricas que no son exclusivas a la GP pero que son fundamentales para esta disciplina. Éste es el caso de las tareas relacionadas con los requisitos, las cuales son en sí mismas un área de la IS de reconocida complejidad e importancia [18]. De allí que es posible hallar soporte específico a la gestión de los requisitos no sólo en una herramienta de GP sino en otras. De tal manera que el peso de estos criterios dentro de la evaluación dependerá de los intereses de la organización y de las prácticas que la caracterizan. La adecuación de los criterios propuestos a las necesidades de las PyMES se encuentra especialmente al nivel de métricas. Éstas consideran necesidades comunes a cualquier proyecto software pero se focalizan en atributos de las herramientas que son especialmente relevantes en el contexto de estas empresas.

4 Aplicación de los criterios y discusión de resultados

Una vez establecidos los criterios, utilizamos cinco herramientas software que ofrecen apoyo a diversas actividades de la GP con el fin de obtener una validación inicial de los mismos. Las herramientas utilizadas para esta validación fueron: (1) *Gantt Project*, (2) *Trac*, (3) *Enterprise Architect (EA),* (4) *DotProject* y 5) *OpenProj*.

Para la selección de estas herramientas, privó en primera instancia que fuesen construidas bajo la filosofía de Software Libre. Sin embargo, además se seleccionó a EA [19] porque provee funcionalidades claramente dirigidas a la GP, y posee también otras funcionalidades dirigidas al Análisis y Diseño. Asimismo, si bien EA se diferencia por ser propietaria, su evaluación facilitó – dado su alcance funcional – identificar debilidades de las restantes en cuanto su aporte específico en tareas relevantes a la GP, como es el caso de los requisitos.

Gantt Project [20], DotProject [21] y OpenProj [22] son herramientas software para el apoyo a la Gestión; Trac [23] puede definirse como una herramienta para el seguimiento de incidentes. Por ello, estas herramientas pueden considerarse como herramientas de apoyo a la Planeación [18], o a la Planificación y Seguimiento [12].

La evaluación de los criterios incluyó la aplicación y verificación de la satisfacción de cada una de las métricas propuestas. Como se indicó anteriormente, los valores asignados a cada métrica corresponden a una escala de medición entre 1 y 5, donde 5 expresa el mejor valor posible (la métrica es satisfecha idealmente por la herramienta) y 1 representa el valor menos favorable (la métrica no es satisfecha por la herramienta). Dentro de esta escala, se consideró satisfecha una métrica en aquellos casos donde el valor fue igual o superior a 3. Para el cálculo de los resultados se totalizó el número y el porcentaje de métricas satisfechas por cada pregunta formulada. Estos resultados se muestran en la Tabla 6.

Por su parte, las figuras 1-3 presentan los porcentajes obtenidos por las herramientas respecto a cada objetivo de evaluación. La Fig. 1 muestra la satisfacción de las herramientas respecto al soporte que proporcionan al Inicio y definición del alcance del proyecto (objetivo 1), calculado con base a las 20 métricas correspondientes a este objetivo.

Tabla 6. Porcentajes de satisfacción obtenidos para las preguntas formuladas y sus objetivos

	Gannt Project	Trac	EA	DotProject	OpenProj
O1: Inicio y definición de alcance	**35,00**	**30,00**	**70,00**	**40,00**	**40,00**
P1: Definición de requisitos	41,67	33,33	75,00	50,00	50,00
P2: Análisis factibilidad	40,00	40,00	40,00	40,00	40,00
P3: Revisión requisitos	0,00	0,00	100,00	0,00	0,00
O2: Planificación	**36,21**	**39,66**	**34,48**	**43,10**	**51,72**
P1:Planificación	22,22	33,33	55,56	22,22	44,44
P2: Definición de Entregables	50,00	66,67	66,67	50,00	50,00
P3: Estimación esfuerzos y costos	48,00	48,00	0,00	60,00	68,00
P4: Asignación de recursos	50,00	50,00	62,50	62,50	75,00
P5: Gestión de riesgos	0,00	0,00	75,00	0,00	0,00
O3: Ejecución	**18,75**	**68,75**	**31,25**	**75,00**	**43,75**
P1: Seguimiento y control	18,75	68,75	31,25	75,00	43,75

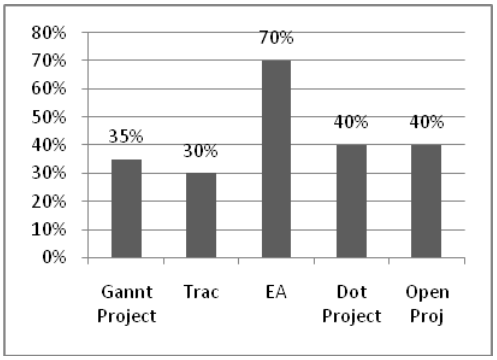


Fig. 1. Porcentaje de satisfacción alcanzado para el Inicio y Definición del alcance

Vemos como EA arrojó el mejor resultado (70%) para el objetivo 1. Al profundizar este análisis con base en los resultados presentados en la Tabla 6, vemos que EA destaca por su soporte a los requisitos, obteniendo 75% y 100% de satisfacción en la Definición y en la Revisión de requisitos, respectivamente. Esto se debe a que EA presenta capacidades para crear requisitos, presentar tipos de requisitos predefinidos, crear nuevos tipos de requisitos, asignar estados a los requisitos y registrar restricciones del proyecto. En segundo lugar le siguen DotProject y OpenProj, las cuales presentan un valor medio para la Definición de requisitos alejado del valor obtenido por EA; al mismo tiempo estas herramientas (al igual que GanntProject y Trac) obtuvieron valores nulos en cuanto a la Revisión de requisitos.

Al analizar globalmente los resultados del objetivo 1, podemos suponer que todas las herramientas, excepto EA, no han considerado la Definición de requisitos como parte de la GP, debido probablemente a la naturaleza general de esta tarea. El hecho de que EA la incluya puede deberse a que esta herramienta intenta apoyar en etapas tempranas del proceso de desarrollo. Esto nos lleva a suponer que el peso que otorgará la organización evaluadora al objetivo 1 dependerá de que ésta cuente ya con alguna herramienta de Gestión de requisitos. Sin embargo, alertamos sobre la conveniencia de que se maneje la integridad entre los requisitos, la planificación y la ejecución.

La Fig. 2 muestra los resultados obtenidos por las herramientas en cuanto al soporte a la Planificación (objetivo 2), calculados en base a 58 métricas. En la gráfica podemos apreciar que OpenProj obtiene el mejor valor, seguida por DotProject.

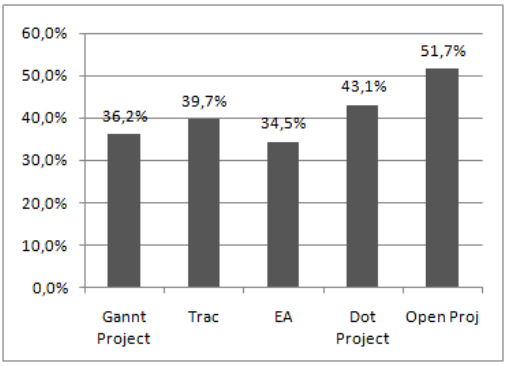


Fig. 2. Porcentaje de satisfacción alcanzado para la Planificación del proyecto software

En un análisis detallado de las preguntas relacionadas con el objetivo 2 (ver la Tabla 6) puede notarse que los resultados difieren entre preguntas. En primer lugar, en cuanto a la Planificación, el resultado de EA (55,56%) es superior a OpenProj (44,44%) y Trac (33,33%). Esto se debe a que EA presenta fases predefinidas asociadas al ciclo de vida del desarrollo. Respecto a la Definición de entregables, Trac y EA ofrecen el mayor soporte (66,67%), en virtud de la flexibilidad para generarlos.

Por otra parte, sobre la Estimación de esfuerzos y costos, OpenProj (68%), seguida de DotProject (60%), arrojaron mejores resultados, principalmente por sus posibilidades de soporte al manejo de tareas y a la calendarización, respectivamente. En cuanto a la Asignación de recursos el mayor soporte es ofrecido por OpenProj (75%), dada la flexibilidad para definirlos y representarlos. Finalmente, aún cuando EA es la única herramienta que no está dirigida explícitamente a la GP, ésta es la única que soporta la Gestión de riesgos (75%), aunque es también la única que no presenta soporte a la Estimación de esfuerzo y recursos.

Desde una perspectiva general, vemos que para el objetivo 2 los resultados se acercan bastante a la realidad, ya que la relación entre EA, DotProject y OpenProj se invierte (EA pasa de ser la herramienta destacada a aquella con peor resultado), lo que parece estar justificado por la naturaleza de estas herramientas. En este punto, resulta necesario recomendar que el peso asignado al objetivo 2 dentro de la evaluación sea significativo, ya que el mismo es uno de los más importantes para la GP.

La Fig. 3 muestra los resultados del soporte provisto por las herramientas a la Ejecución del proyecto (objetivo 3), calculado en base a las 16 métricas correspondientes. Nuevamente obtenemos un resultado ligeramente diferente a los anteriores, ya que DotProject se destaca como la herramienta con el mayor valor de satisfacción (75%); al mismo tiempo que Trac ocupa el segundo lugar con un 68,75%. Estos resultados reflejan el soporte de las herramientas a actividades de seguimiento y control, principalmente como consecuencia de la generación de vistas, recordatorios y filtros de tareas. Estas herramientas soportan, asimismo, el manejo de incidentes y acciones correctivas, una capacidad también provista por EA.

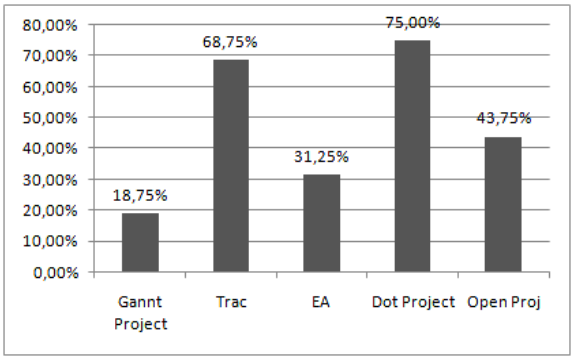


Fig. 3. Porcentaje de satisfacción alcanzado para la Ejecución del proyecto software.

De la aplicación de las métricas, también se desprende que Gantt Project, Trac y EA presentan debilidades sobre el control de horas planificadas y trabajadas, así como sobre los recursos planificados y los ejecutados.

En general, pensamos que estos resultados son coherentes con la clasificación que asignamos a las herramientas evaluadas; asimismo, parece aconsejable que el objetivo 3 cuente con un peso relevante dentro de la evaluación, ya que el Seguimiento y control de las tareas planificadas es una actividad fundamental para la GP.

A partir de los resultados anteriores podemos obtener conclusiones relacionadas con el comportamiento de las métricas y de las herramientas evaluadas. En cuanto a lo primero, ha sido posible aplicar las métricas directamente identificando fortalezas y debilidades de las herramientas software que son coherentes con naturaleza de las mismas. Respecto al comportamiento de las herramientas observamos que aún cuando están dirigidas a apoyar la GP, pueden existir marcadas diferencias entre las actividades que apoyan. De manera resumida tenemos que:

- EA destaca por su soporte en cuanto al Inicio y definición del alcance del proyecto, debido a su apoyo a los requisitos. Al mismo tiempo, presentó una bondad importante relacionada con Fases predefinidas del ciclo de vida de desarrollo y la flexibilidad en la Generación de entregables. Por otra parte, se destaca en cuanto al Manejo de incidentes y acciones correctivas, y resultó la única herramienta evaluada que soporta explícitamente la Gestión de riesgos.
- Trac demostró flexibilidad en la Generación de entregables y fortalezas para el Seguimiento y control gracias a las vistas, recordatorios y filtros de tareas.
- OpenProj ofreció excelente soporte al Manejo de tareas y a la Calendarización, al mismo tiempo que obtuvo los mejores resultados respecto a la Asignación de recursos, debido a su flexibilidad para definirlos y representarlos.
- GanntProject, aunque presenta valores cercanos a las otras herramientas para casi todos criterios, no logra destacar respecto a ninguna característica. Cabe resaltar que presenta, con diferencia, el peor soporte en cuanto al Seguimiento y control.
- Finalmente, DotProject demostró fortalezas para el Seguimiento y control mediante la generación de vistas, recordatorios y filtros de tareas, el Manejo de tareas, la Calendarización, y el Manejo de incidentes y acciones correctivas.

Además de estas particularidades, las herramientas evaluadas presentaron algunas fortalezas comunes entre todas ellas, las cuales se refieren a la Creación de proyectos,

Manejo de tiempo y Vistas del personal. Al mismo tiempo, en la Tabla 6 podemos observar que ciertas métricas arrojaron valores nulos en la mayoría de las herramientas evaluadas. Éstas se encuentran relacionadas con aquellas preguntas dirigidas a la *Revisión de requisitos* y la *Gestión de riesgos*; lo que nos lleva a suponer que estas actividades podrían estar siendo consideradas fuera del alcance de las GP por parte de los desarrolladores de estas herramientas, a pesar de ser áreas muy relevantes en el contexto de las PyMES, lo que puede convertirse en una oportunidad para futuras evaluaciones. Por último, tal y como hemos mencionado anteriormente, la selección de alguna de estas herramientas dependerá del peso que otorgue la organización evaluadora a cada uno de los objetivos de evaluación.

5 Conclusiones y trabajo futuro

Para evaluar y seleccionar herramientas software de apoyo a la GP se requiere de ejercicios contextuales que incluyan aspectos de la empresa involucrada y consideren el propio entorno de la IS. En el caso de las PyMES, las actividades relacionadas con la estimación de tareas, la asignación de recursos, el seguimiento y control, entre otras actividades presentadas en este trabajo, constituyen necesidades que derivan de la realidad de estas empresas y que requieren apoyo de las herramientas de IS.

En esta investigación, hemos logrado la efectividad del ejercicio de evaluación y selección al determinar los niveles de satisfacción de criterios que representan las necesidades de las PyMES en el ámbito de la GP. Tales criterios han sido propuestos con base en objetivos claves de la GP, tales como el soporte al Inicio y definición del alcance del proyecto, el soporte a la Planificación y el soporte a la Ejecución del mismo. Hemos propuesto y aplicado un conjunto de 92 métricas a un conjunto de herramientas representativas de la práctica actual, lo cual nos ha permitido comprobar la generalidad y la viabilidad de nuestro enfoque. Además, como resultado de la aplicación de estas métricas, hemos identificado las bondades y limitaciones de las herramientas evaluadas. De esta manera, pretendemos proporcionar a los profesionales de la IS un enfoque completo, coherente y repetible para la toma de decisiones, así como un conjunto de recomendaciones en torno a las herramientas que hemos analizado; destacando la necesidad de asignar pesos a los objetivos de evaluación establecidos, con base en los intereses de la organización evaluadora.

Trabajos futuros se orientarán a la propuesta y refinamiento de métricas, al desarrollo de estudios de campo y a la elaboración de instrumentos orientadores, hasta la completa formulación de un modelo dirigido a PyMES, que facilite la evaluación y selección de herramientas de apoyo a diversas áreas de la IS.

Agradecimientos. Esta investigación ha sido financiada por el Fondo Nacional para la Ciencia, Tecnología e Innovación (FONACIT) de Venezuela a través del proyecto G-2005000165.

Referencias

1. Fox, T., Spence, W: The Effect of Decision Style on the Use of a Project Management

Tool: An Empirical Laboratory Study, The DATA BASE for Advances in Information Systems, vol. 36, no. 2, 28-42 (2005)
2. Murphy, A., Ledwith, A .:Project management tools and techniques in high technology SMES. Management research news, vol. 30, no. 2,153-166 (2007)
3. Lundell, B., Lings, B.: Comments on ISO 14102: the standard for CASE-tool evaluation. Computer Standards & Interfaces, vol. 24, 381–388 (2002)
4. Kitchenham, B.: Evaluating software engineering methods ant tools. Part 1: The evaluation context and evaluation methods. ACM SIGSOFT Software Engineering Notes, vol. 21, no. 1, 11–14 (1996)
5. ISO/IEC: Information Technology—Guideline for the evaluation and selection of CASE tools. ISO/IEC JTC1/SC7. ISO/IEC 14102:1995(E). USA (1995)
6. IEEE: IEEE Recommended Practice for the Evaluation and Selection of CASE Tools. IEEE Std 1209-1992. Published by the Institute of Electrical and Electronics Engineers, USA (1993)
7. O'Connor, R., Baddoo, N., Smolander, K., Messnarz, R.: Software Engineering Lifecycle Standard for Very Small Enterprises. Communications in Computer and Information Science. Software Process Improvement 15th European Conference, EuroSPI 2008, Dublin, Ireland, September 3-5 (2008).
8. O'Connor, sv-lncs. Version 1.2. S. Editors: ALEXANDRE – Centre d'Excellence en Technologies de l'Information et de la Communication (CETIC), (Belgium); C. Y. LAPORTE – Ecole de Technologie Supérieure (ETS), (Canada) . Estatus: draft (2007)
9. Rivas, L., Pérez, M., Mendoza, L. & Grimán, A. Selection criteria for software development tools for SMEs, Internacional Conference on Enterprise Information Systems, Barcelona, España (2008)
10. Basili,V., Gianluigi, C., H. Dieter: Goal Question Metric Paradigm. Encyclopedia of Software Engineering, vol. 1, edited by John J. Marciniak, John Wiley & Sons, 528-532, (1994)
11. PMI: A Guide to the Project Management Body of Knowledge (PMBOK Guide). Project Management Institute (2000)
12. IEEE: Guide to the software engineering body of knowledge SWEBOK. A project of the IEEE Computer Society (2004)
13. Fuller, M., Valacich, J., George, J.: Information systems project management: a process and team approach. Pearson education. USA (2008)
14. Richardson, I., Gresse, C.: Why are small software organization different?, IEEE software, published by the IEEE computer society, January/February, 18-22 (2007)
15. Fayad, M., Laitinen, M., Ward, R. Software engineering in the small, Communications of the ACM, vol. 23, no. 3, 115-118 (2000)
16. El Emam, K.: Multi-Method Evaluation of the Practices of Small Software Projects, Proc. First International Research Workshop for Process Improvement in Small Settings, Pittsburgh, PA, 12-17 (2005)
17. Laporte, C., April. A., Renault, A.: Applying ISO/IEC Software Engineering Standards in Small Settings: Historical Perspectives and Initial Achievements. Proceedings of SPICE 2006 Conference, May 4-5, Luxembourg (2006)
18. Sommerville, I. Ingeniería del Software. 7ma. ed. España: Pearson Educación (2005)
19. Sparx Systems. Enterprise Architect. Recuperado el 25/04/2009 de: http://www.sparxsystems.com.au/products/ea/index.html
20. Gantt Project. Recuperado el 25/04/2009 de: http://www.ganttproject.biz/
21. DotProjet. Recuperado el 25/04/2009 de: http://www.dotproject.net/
22. OpenProj. Recuperado el 25/04/2009 de: http://openproj.org/openproj
23. Trac Open Source Project. Recuperado el 25/04/2009 de: http://trac.edgewall.org/

TEMPO: Una herramienta para la reutilización efectiva en la ingeniería de procesos software

Orlando Avila-García, Adolfo Sánchez-Barbudo Herrera, Víctor Roldán Betancort, Carlos González Pérez, Antonio Estévez García

Open Canarias, S.L.
C/. Elías Ramos González, 4 - Oficina 304
38001 Santa Cruz de Tenerife, Spain
{orlando,adolfosbh,vroldan,caglezp,aestevez}@opencanarias.com
http://www.opencanarias.com

1. Introducción

La ingeniería de procesos software se ha convertido en una de las más prometedoras áreas de la ingeniería del software, que ofrece técnicas y métodos de control en la definición, personalización y ejecución de los procesos software de una empresa de desarrollo. Uno de los elementos clave en la gestión de los procesos software es la reutilización: actualmente es comúnmente aceptado que no hay dos proyectos de software iguales, que los procesos en una empresa de software comparten similitudes pero presentan las suficientes diferencias como para que su gestión sea una tarea compleja.

SPEM (Software & Systems Process Engineering Meta-Model Specification) [1] es la iniciativa que OMG propone para el modelado formal de los procesos software, como estándar a implementar por las herramientas de diseño de procesos con el fin de facilitar la interoperabilidad entre las mismas. Aunque SPEM centra una parte importante de su contenido en proporcionar mecanismos para la reutilización de modelos de procesos, creemos que éstos aportan una complejidad excesiva a la creación de los procesos al seguir unas pautas y/o mecanismos de reutilización tradicionales y comunes a los lenguajes de programación de software, que a lo largo de los años han demostrado sus limitaciones, sobretodo para usuarios no expertos en técnicas de programación. En trabajos previos ya se habla sobre limitaciones presentes en los mecanismos de reutilización de SPEM [2].

En [3] se propone el concepto de ingeniería de líneas de proceso software, como un método de reutilización efectiva en el diseño de procesos software que utiliza conceptos y técnicas de la llamada ingeniería de líneas de producto. La ingeniería de líneas de producto software permite la reutilización efectiva de artefactos de software basándose en la identificación, organización y gestión proactivas de artefactos de software similares para un dominio dado, acorde con sus similitudes y variaciones. Recientemente han comenzado a aparecer investigaciones sobre la definición de estas líneas de procesos software. Por ejemplo, [2] propone una extensión de SPEM 2.0 para modelar explícitamente la variabilidad en líneas o familias de procesos.

Sin embargo, el proceso de creación de una línea de producto va más allá de la simple definición de las características comunes y variables a los productos de la familia,

Actas JISBD 2009, pp.81-84, ISBN: 978-84-692-4211-7

en la llamda ingeniería de dominio. Otras actividades son igualmente importantes, como la ingeniería de aplicación, que trata la derivación de un producto concreto a partir de una selección (configuración) particular de características, y la llamada gestión de la línea de productos, que se encarga del control estratégico y coordinación de los esfuerzos aplicados a las dos anteriores disciplinas. TEMPO es una herramienta que pretende dar un soporte integral a la ingeniería de líneas de proceso software, permitiendo no solo la definición de familias de procesos (ingeniería de dominio) sino la configuración y derivación de los procesos concretos de esas familias (ingeniería de aplicación), y la coordinación y gestión de la interacción entre ambos esfuerzos.

2. Definición de procesos genéricos y concretos.

En TEMPO identificamos dos roles bien diferenciados, el llamado “diseñador de procesos” y el “configurador de procesos”. La herramienta ofrece una perspectiva de trabajo para cada uno de estos roles, configurando el workbench para mostrar la información más relevante para cada rol. El diseñador es el encargado de crear la familia de procesos SPEM a través del denominado proceso genérico, modelando las características comunes y puntos de variabilidad de los procesos de la familia. Esta actividad es la que denominamos típicamente ingeniería de dominio en la ingeniería de líneas de producto software.

El configurador realiza actividades de configuración sobre procesos genéricos para obtener las instancias concretas de procesos SPEM que pertenecen a la familia. Para ello, y siguiendo un asistente de configuración proporcionado por la herramienta, el configurador selecciona entre las características variables (opcionales y alternativas) de la familia; esta selección se denomina configuración y será utilizada por la herramienta para resolver automáticamente la variabilidad del proceso genérico y producir el proceso concreto. Esta es la actividad que se corresponde con la llamada ingeniería de aplicación en la ingeniería de líneas de producto software.

En TEMPO, el diseñador de procesos genéricos utiliza un lenguaje que permite definir, por un lado, los elementos SPEM del proceso que formarán parte de los procesos concretos de la familia, y por el otro, elementos de variabilidad que completarán la definición de la familia, estableciendo la presencia obligatoria, opcional o alternativa de dichos elementos SPEM en los procesos de la familia. Este lenguaje se llama MOTEL (Model Template Language) y ha sido inspirado en el modelo de características de Czarnecki [4], que permite modelar varibilidad en líneas de producto software. MOTEL ha sido diseñado de tal manera que sus instancias mantienen embebidos los elementos SPEM que formarán parte de los procesos de la familia. De esta manera, MOTEL puede ser interpretado como un decorador del lenguaje SPEM, que añade información de variabilidad sobre los elementos de dicho lenguaje.

Para facilitar la interacción del usuario final con dicho lenguaje, TEMPO ofrece una sintaxis concreta gráfica inspirada en la notación UML. Un editor GMF (Graphical Modeling Framework) ayuda al diseñador a crear dichos procesos genéricos de forma que se utilizan cajas para representar las características variables del proceso, que a su vez contendrán los elementos SPEM que las realizan. En la siguiente figura podemos observar un ejemplo de dicha sintaxis:

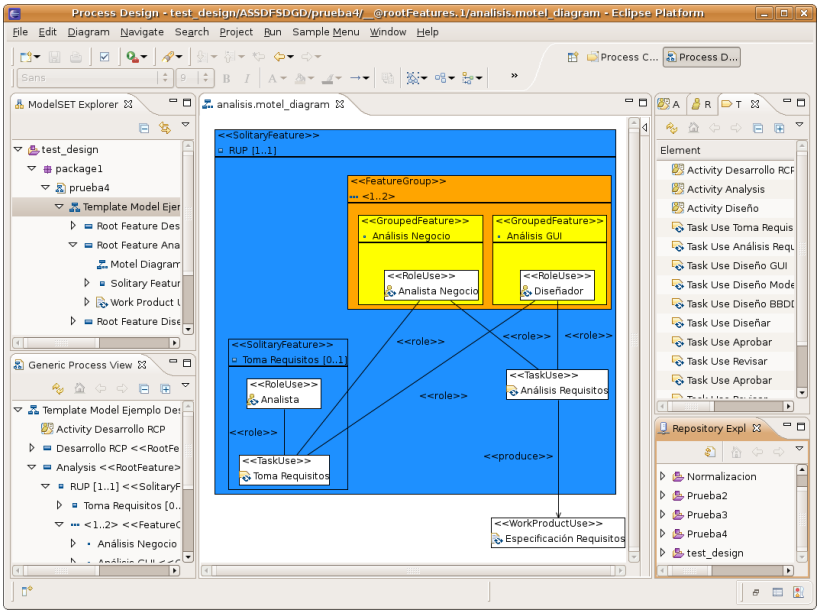


Figura 1. Perspectiva del diseñador de procesos genéricos, editando un modelo MOTEL con notación UML. MOTEL es una combinación de elementos SPEM y elementos de definición de variabilidad.

Cuando el configurador genera un proceso concreto a partir de la configuración del proceso genérico, obtiene un modelo SPEM en el que ya no tenemos elementos MOTEL relativos a la información de variabilidad. El usuario podrá introducir información adicional para especificar más si cabe su proceso concreto.

Finalmente decir que la herramienta guarda la información de configuración del proceso genérico y la mantiene junto al proceso concreto generado, para validar en el futuro que el proceso concreto sigue siendo consecuente con la configuración de la que partió. Esto se hace para controlar la calidad del proceso software incluso después de haberlo configurado, de forma que se puede utilizar para validar que cualquier modificación posterior realizada directamente sobre el proceso concreto cumple con la configuración realizada sobre el proceso genérico original.

3. Publicación y consumo de activos reutilizables

El concepto de artefacto reutilizable es muy común en los entornos de desarrollo de software, cada uno con su propia manifestación de la misma idea: procedimientos, objetos, componentes, aspectos, servicios, etc. La OMG ofrece un modelo formal para especificar activos reutilizables, independiente de la tecnología usada, conocido como RAS (Reusable Asset Specification) [5], que presenta una abstracción conveniente para describir artefactos reutilizables: un paquete que describe tanto qué puede ser reutilizado como la forma de hacerlo, definiendo explícitamente tanto los punto de variabilidad

de los artefactos a reutilizar como el proceso por medio del cual configurarlos. Inspirado en este modelo, TEMPO define una abstracción sencilla que describe conceptos tales como activo reutilizable, servicios, contenedores de artefactos, repositorios, etc., a través de interfaces Java. Usando este modelo, la herramienta es capaz de publicar e importar procesos genéricos creados por un diseñador. El proceso de publicación empaqueta en el activo no solo la descripción del proceso, sino un conjunto de servicios que permitirán mas tarde cargar este activo en otra instancia de TEMPO para ser configurado y reutilizado.

Como hemos explicado, en TEMPO se crean procesos genéricos que deben ser distribuidos para su uso por parte de distintos configuradores. Todo este proceso de creación y publicación del activo se hace de manera transparente al usuario. La herramienta ofrece asimismo facilidades para que el configurador de procesos explore repositorios de activos reutilizables en busca del proceso genérico deseado, y para que lo desempaquete y configure una vez encontrado. Una vez más, la parte más pesada del proceso de consumo del activo es llevada a cabo por la herramienta. Este proceso de publicación/consumo de activos se correspondería con la parte más pesada de la gestión de activos reutilizables de la ingeniería de líneas de producto software. De esta manera, TEMPO completa el soporte integral a los tres procesos para una reutilización efectiva.

Referencias

1. OMG: Software & Systems Process Engineering Meta-model specification, v2.0. Technical Report formal/2008-04-01, OMG (April 2008) Available at http://www.omg.org/docs/formal/08-04-01.pdf.
2. Martínez-Ruiz, T., García, F., Piattini, M.: Towards a spem v2.0 extension to define process lines variability mechanisms. Software Engineering Research, Magament and Applications **150** (2008) 115–130
3. Rombach, D.: Integrated software process and product lines. Unifying the Software Process Spectrum **3840** (2006) 25–31
4. Czarnecki, K., Helsen, S., Eisenecker, U.: Staged configuration through specialization and multi-level configuration of feature models. Software Process Improvement and Practice, special issue on Software Variability: Process and Management **10**(2) (2005) 143–169
5. OMG: Reusable asset specification. Technical Report ptc/04-06-06, OMG (Jun 2004) Available at http://www.omg.org/docs/ptc/04-06-06.pdf.

Una Aplicación Web de Gestión Empresarial Orientada a Proyectos para PYMEs*

Leticia González Folgueira, Miguel R. Luaces

Laboratorio de Bases de Datos, Universidade da Coruña
Campus de Elviña, 15071, A Coruña, España
{lgonzalezf,luaces}@udc.es

Resumen En la actualidad es casi imprescindible disponer de una herramienta informática que permita almacenar y gestionar toda la información generada por la actividad de una empresa. En este trabajo presentamos una herramienta web que permite realizar las principales tareas de gestión empresarial. Se ha optado por una aplicación web para facilitar su uso en entornos distribuidos proporcionando una mayor accesibilidad que una aplicación clásica orientada a escritorio. El núcleo central de la aplicación se centra en la gestión de proyectos que incluye la planificación de tiempos, costes, recursos y esfuerzos, el control de los proyectos y sus actividades y observar la evolución de los mismos en base al tiempo, coste y esfuerzo y las correspondientes comparativas entre lo estimado y lo real. En la administración empresarial también se incluye la gestión del personal de la empresa, el mantenimiento de una agenda de clientes y contactos y la gestión de recursos y reservas de los mismos.

1. Motivación

Hoy en día, realizar la gestión de proyectos es un trabajo difícil para empresas ya que se ven inmersas en un montón de papeles expandidos con notas sobre las tareas a realizar, la planificación de costes, etc. Cualquier empresa necesita conocer diversos datos sobre sus proyectos para poder controlar presupuestos, gastos, ingresos y beneficios. De hecho, los beneficios son el sustento de toda empresa, por lo cual es imprescindible una correcta gestión para lograr los objetivos satisfactoriamente.

Este proceso de gestión sería complejo e implicaría muchas horas de trabajo sin la ayuda de una herramienta informática, mientras que es un proceso sencillo con una aplicación que facilite el seguimiento de los proyectos y la comparación de todos los datos anteriormente señalados. Además, la información de esta herramienta puede servir para estimar costes de proyectos similares que se afrontarán en un futuro próximo. El control de los proyectos implica, además de lo mencionado, conocer el tiempo que dedica cada empleado a cada tarea y el coste asociado a dicho trabajo que repercute directamente en los costes del proyecto. Además, la disponibilidad de esta información permite también evaluar el rendimiento del empleado.

Por lo tanto, es clara la necesidad de disponer de una herramienta sencilla y eficaz que facilite la gestión empresarial. Por estos motivos, el Laboratorio de Bases de Datos

* Este trabajo ha sido financiado parcialmente por el Ministerio de Educación y Ciencia (PGE y FEDER) ref. TIN2006-15071-C03-03 y por la Xunta de Galicia ref. 2006/4 y ref. 08SIN014E.

de la Universidad de A Coruña decidió desarrollar una aplicación web que proporcione de forma intuitiva las funcionalidades imprescindibles para la gestión empresarial.

2. Funcionalidad

Se ha optado por una herramienta web ya que así facilita la gestión empresarial en entornos colaborativos, es decir, donde las personas pueden estar físicamente en distintas localizaciones pero necesitan compartir y acceder a la información en tiempo real. De esta forma, se consigue dotar a la aplicación de mayor accesibilidad que una aplicación clásica. Principalmente, la herramienta proporciona las siguientes funcionalidades:

- Gestión de distintos perfiles de usuario de tal forma que el uso que una persona puede hacer de la aplicación depende de los permisos de su rol de usuario.
- Gestión del personal de la empresa y los usuarios asociados que permiten el acceso a la aplicación.
- Gestión de proyectos y sus actividades. Este proceso incluye la planificación de tiempos y asignación de recursos humanos a las tareas. Además permite realizar el seguimiento de los proyectos y el control económico que se centra en los presupuestos, gastos y facturas asociados a cada proyecto.
- Gestión de clientes y contactos asociados.
- Gestión de recursos que permite controlar el uso de los mismos mediante las reservas que una persona realiza de un recurso en unas fechas determinadas.
- Configuración de ciertos aspectos de la aplicación que dependen del área de trabajo de la empresa, como tipos de gastos, estados de factura, estados de proyecto, tipos de recursos y los departamentos de los que consta la organización.

2.1. Gestión de personal

Los usuarios con rol de administrador son los encargados de la gestión de usuarios y personal. Se almacena toda la información relativa al empleado tal como datos personales, datos laborables, datos fiscales y bancarios y demás información relacionada con su labor en la empresa como permisos y vacaciones, cargos actuales, méritos, formación recibida y currículum correspondiente.

Cuando un empleado está dado de alta se le puede asignar uno o más usuarios con el rol adecuado a su labor en la empresa pudiendo ser: administrador, gerente, responsable de calidad y medio ambiente, responsable de departamento, jefe de proyecto o empleado. De esta forma se restringe la información a las personas no autorizadas y al mismo tiempo facilita el acceso a las funciones principales del perfil según sus permisos y responsabilidades.

Tanto los usuarios con rol de administrador o con rol de gerente pueden acceder a toda la funcionalidad que ofrece la aplicación. Los usuarios con rol de jefe de proyecto o con rol de responsable de departamento se encargan de la gestión de los proyectos que dirigen o que pertenecen a su departamento, respectivamente. Los usuarios con rol de empleado tienen como función principal en la herramienta cubrir sus partes horarios con las horas de esfuerzo dedicado diariamente a las tareas que le fueron asignadas. Finalmente, el rol de responsable de calidad y medio ambiente se define en la norma ISO 9001 y es de especial importancia en empresas que cumplen este estándar.

2.2. Gestión de proyectos

La herramienta desarrollada permite realizar las tareas básicas de la gestión de proyectos. Un proyecto a lo largo de su ciclo de vida pasa por distintos estados que es necesario conocer para el correcto control y seguimiento, por lo cual en la mayoría de las ocasiones la información relativa a los proyectos se muestra agrupada por el estado actual en el que se encuentran.

Este proceso comienza cuando un responsables de departamento da de alta un nuevo proyecto definiendo sus datos generales, como el nombre, la descripción, las fechas estimadas, presupuesto, el cliente del proyecto y el departamento de la empresa encargado del mismo, entre otros. A continuación, se asigna uno o más jefes de proyecto responsables del mismo y a partir de ese momento ambos roles podrán encargarse de la planificación y seguimiento del proyecto.

Entre las tareas principales destacamos la división del proyecto en actividades y subactividades, definiendo las fechas de comienzo y fin estimadas que luego serán completadas con las fechas reales en que se desarrolló dicha actividad. Los recursos humanos se asignan a las actividades concretas indicando las fechas en que se debe realizar el trabajo y la estimación en horas de esfuerzo que implica. Todo usuario dispone de su parte de trabajo en el cual consulta las actividades que tiene asignadas e introduce las horas dedicadas diariamente a cada una de ellas. De esta forma la información de los proyectos se actualiza instantáneamente.

Para un mayor detalle económico se controlan los gastos y facturas de cada proyecto, detallando el tipo de gasto en cuestión y el importe del mismo, y en el caso de las facturas las fechas relevantes, el importe y el estado en que se encuentra. Con todo esto, se permite al responsable de departamento y al jefe de proyecto consultar en todo momento los datos actualizados en base a tres ejes: económico, tiempo y esfuerzo. Los responsables de los proyectos pueden consultar y comparar todos los datos, obteniendo así información tanto de los costes y beneficios inicialmente estimados como de los que se están obteniendo. Esta comparación se puede hacer también para las fechas estimadas y fechas reales, y para el trabajo inicialmente previsto y las horas reales. Para facilitar esta tarea se dispone de varios listados. El listado principal muestra el resumen de la información económica en base a costes y beneficios estimados en comparación con los obtenidos, y también el esfuerzo planificado frente al trabajo real que finalmente conlleva. Estos resultados se pueden obtener generando dinámicamente un informe en formato PDF.

La aplicación permite desglosar esta información según el detalle deseado, para eso ofrece distintos listados según lo que se desee consultar, como ejemplo destacamos los listados que presentan el trabajo realizado en cada proyecto a lo largo del tiempo, con tablas semanales, mensuales o anuales. Con tablas temporales también se puede conocer la dedicación de cada empleado a sus proyectos, o por el contrario que personas han trabajado en cada proyecto y actividades en determinado día o mes.

2.3. Gestión de recursos y contactos

Generalmente una empresa cuenta con un conjunto de recursos materiales que serán utilizados por los empleados en el momento que sea preciso. Para mejorar organización,

se agrupan por tipo de recurso que dependerá de la actividad que desarrolle la empresa. El control de recursos se lleva a cabo mediante reservas. Un recurso se reserva por una persona entre unas fechas determinadas de forma que ya nadie podrá solicitar ese recurso en dichas fechas y se puede conocer quien es el responsable del mismo en cada momento.

Todo usuario puede consultar los recursos existentes y la lista de sus reservas, pudiendo realizar una nueva reserva siempre que el recurso solicitado se encuentre disponible en la fecha deseada. Con respecto a esta funcionalidad podemos destacar la existencia de un calendario de reservas que muestra para un recurso cuando está reservado y por quién, y el listado que muestra la misma información pero para el conjunto total de recursos pudiendo filtrar por tipo.

Por otra parte, la aplicación permite mantener una agenda de contactos que generalmente van asociados a un cliente concreto. Esta agenda se divide en contactos públicos (que puede consultar cualquier usuario) o privados (a los que sólo tiene acceso la persona que creó el contacto y los administradores).

3. Conclusiones

Se ha desarrollado una aplicación web que facilita la gestión de proyectos en entornos distribuidos mediante interfaces sencillas que proporcionan comodidad y facilidad de acceso a las funciones más necesarias en cada página de la aplicación. La aplicación implementa las funcionalidades básicas de la gestión de proyectos con información siempre actualizada al ir imputando horas de esfuerzo mediante partes horarios. Además, la aplicación incluye control de gastos y facturación, gestión de recursos, agenda de contactos y clientes, soporte multilingüe y facilidad de configuración adaptando la herramienta a la actividad que se desarrolla en la empresa.

Por lo tanto, consideramos que el uso de la herramienta facilita la administración y gestión de una PYME.

Parte IV

Sesión 3. Requisitos / Ingeniería del software empírica

Pautas para Agregar Estudios Experimentales en Ingeniería del Software

Enrique Fernández, Oscar Dieste, Patricia Pesado, Ramón García-Martínez

Programa de Doctorado en Ciencias Informáticas. Facultad de Informática. UNLP. Argentina
Grupo de Ingeniería de Software Experimental. Facultad de Informática. UPM. España
Instituto de Investigaciones en Informática LIDI. Facultad de Informática. UNLP - CIC
Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. UBA. Argentina
Área Ingeniería del Software. Licenciatura en Sistemas. Departamento de Desarrollo Productivo y Tecnológico. UNLa. Argentina

Resumen. El presente trabajo muestra cómo la aplicación de métodos de Meta-Análisis alternativos al método Diferencias Medias Ponderadas, como son el Response Ratio o el Conteo de Votos, puede ayudar a mejorar las piezas de conocimientos generadas en las Revisiones Sistemáticas hechas dentro del campo de la Ingeniería del Software.

1. Introducción

La agregación de estudios experimentales en Ingeniería del Software (IS) fue propuesta originalmente por [1] en 1996 y a partir de la presentación del trabajo de [2] en 2004 el interés por su desarrollo por parte de la comunidad científica se incrementa considerablemente [2; 4; 5; 6; 7]. Según [8] la agregación de experimentos consiste en combinar los resultados de varios estudios experimentales, previamente desarrollados, que analizan el comportamiento de un par de tratamientos específicos, en un contexto determinado con el objeto de generar nuevas piezas de conocimiento. Cuando este tipo de trabajo es realizado mediante métodos estadísticos, se los suele llamar Meta-Análisis (MA), término empleado por primera vez por [9]. Si bien el MA aporta un conjunto de herramientas para poder combinar de manera fiable los resultados de los estudios, para que las conclusiones aportadas sean consideradas representativas no debe existir sesgos en la selección de los estudios que se van a agregar. Esto se logra mediante el uso de las Revisiones Sistemáticas (RS). Una RS es un procedimiento que aplica estrategias científicas para aumentar la fiabilidad del proceso de recopilación, valoración crítica y agregación de los estudios experimentales relevantes sobre un tema [10]. Si bien, en la actualidad se conocen muchas RS en el ámbito de la IS que lograron recopilar experimentos y hacer una valoración crítica de los mismos. Estas, en general, fallan a la hora de agregar los resultados, principalmente, por no poder aplicar el método Diferencias Medias Ponderadas [11] (DMP) propuesto en [2]. Así por ejemplo, en [3] no se pudo desarrollar el MA debido a la escasez de replicaciones y la baja calidad de los informes publicados, o en [4] no se puedo desarrollar el MA debido a la gran diversidad de técnicas y variables respuesta utilizadas. Este hecho motivó que

Actas JISBD 2009, pp.91-102, ISBN: 978-84-692-4211-7

algunos autores viendo la dificultad de desarrollar un MA, debieron recurrir a métodos alternativos no estadísticos. Por ejemplo en [3] se realizó una sumarización de los estudios a favor de cada tratamiento, y proclamó como ganador al tratamiento con mayor cantidad de estudios a favor, lo cual constituye una instancia de un procedimiento bien conocido en Ciencias Sociales y denominado Conteo de Votos no Estadístico [12] (CVNE). Si bien de esta forma es posible derivar nuevas piezas de conocimiento de tipo general, la fiabilidad de las mismas está en entredicho ya que el conteo de votos tiende a producir falsos resultados negativos cuando el poder estadístico de los test es reducido [11]. Ello puede provocar que muchas piezas de conocimiento sean erróneas. Con el objeto de mitigar este problema se han desarrollado nuevos métodos de MA, que sean menos restrictivos en cuanto a sus condiciones de aplicación, pero que mantengan la esencia de fiabilidad que los métodos estadísticos deben aportar.
El objetivo de este trabajo es presentar dos métodos de MA alternativos a DMP, como son el Conteo de Votos Estadístico (CVE) y el Response Ratio (RR), y mostrar como la fiabilidad de las piezas de conocimiento aumenta frente a lo que obtenemos con el CVNE. En la sección 2 de este artículo se describen los métodos de agregación; en la sección 3 se describen un conjunto de pautas respecto de cómo y cuándo utilizar los distintos métodos; en la sección 4 se presenta un caso de estudio; en la sección 5 se presenta un conjunto de puntos de discusión; finalmente, en la sección 6 se describe algunas de las conclusiones obtenidas.

2. Estado del Arte

2.1 Agregación de Estudios Experimentales

Si todos los estudios a incluir en un proceso de agregación fuesen igualmente precisos, bastaría con promediar los resultados de cada uno de ellos para obtener así una conclusión final. Sin embargo, en la práctica no todos los estudios tienen la misma precisión, por ello cuando se los combine se debe asignar un mayor peso a los estudios que permiten obtener información más fiable. Esto se logra combinando los resultados mediante un promedio ponderado [13]. Además como los resultados de los diferentes estudios pueden medirse en diferentes escalas de la variable respuesta [11], la variable dependiente en un MA debe poder compatibilizar estos aspectos, lo cual se logra mediante la estimación de un "tamaño de efecto", el cual consiste en un estimador estandarizado no escalar de la relación entre una exposición y un efecto. En sentido general, este término se aplica a cualquier medida de la diferencia en el resultado entre los grupos de estudio (por ejemplo, cantidad de líneas de código que posee un programa o cantidad de requisitos educidos). Cuando lo que se desea combinar son estudios que trabajan con variables continuas (como sucede en IS), el método a aplicar por excelencia es Diferencias Medias Ponderadas [1; 2]. Este método es conceptualmente sencillo [9]: el tamaño de efecto de cada estudio se estima como la diferencia de la medias dividida por la varianza conjunta de ambos tratamientos:

$$g = \frac{Y^E - Y^C}{S_P}$$

g es el tamaño de efecto
Y's son las medias de los tratamientos
S_p es la varianza conjunta **(1)**

La función (1) fue optimizada por [11], quien incorporó un factor de corrección "J" que se utiliza para aumentar la fiabilidad del método cuando los estudios a agregar cuentan con pocos sujetos experimentales. La nueva función se la conoce como "*d*" y es la recomendada en [2; 8]. Una vez estimado el tamaño de efecto de cada estudio se debe estimar el tamaño de efecto general o global, para ello se utiliza la siguiente función:

$$d^* = \frac{\sum d_i / \sigma^2{}_i(d)}{\sum 1/\sigma^2{}_i(d)}$$

d^* es el tamaño de efecto global
$\sum d_i / \sigma^2{}_i(d)$ es la suma de los efectos individuales
$\sum 1/\sigma^2{}_i(d)$ es la suma de la inversa varianza **(2)**

Para mayores detalles de cómo aplicar las formulas indicadas remitirse a [11]. Es importante destacar que para que este método pueda aplicarse el informes del experimento debe indicar: número de sujetos, medias y varianzas, lo cual en general no ocurre en los artículos presentados en IS [14], por ello muchos autores han recurrido a alternativas de agregación no estadísticas. Es de hacer notar que si bien existen algunas versiones de DMP alternativas a las indicadas anteriormente en las cuales el tamaño de efecto es estimado a partir de los parámetros F o t [11], estas funciones no se encuentran muy difundidas y se desconoce si las mismas arrojan resultados fiables o no.

2.2 Métodos Alternativos de Agregación

2.2.1 Conteo de Votos estadístico

Este método fue propuesto como una alternativa a DMP por [11] para agregar estudios con carencias en los informes o cuando se requiere combinar estudios que utilizan variables respuesta diferentes, pero, compatibles entre sí (por ejemplo estudios que miden el tamaño del código de un programa a través de la cantidad de clases podrían juntarse con otros que miden el tamaño en función de la cantidad de líneas de código). Su principal ventaja radica en que para estimar el tamaño de efecto solo se requiere conocer si existe o no diferencia entre las medias de los tratamientos y la cantidad de sujetos experimentales. Con esta información se realiza un proceso de inferencia que intenta determinar cuál es el tamaño de efecto de mayor probabilidad de ocurrencia en base a los parámetros dados. Su función principal es la siguiente:

$$L(\delta \mid X_1, ..., X_i) = \sum_{i=1}^{k} \left\{ \begin{matrix} X_i \ln\left[1 - \phi\left(-\sqrt{\tilde{n}}\delta\right)\right] + \\ (1 - X_i)\ln \phi\left(-\sqrt{\tilde{n}}\delta\right) \end{matrix} \right\}$$

$L(\delta|X_1, ...X_n)$ es la probabilidad de tamaño de efecto
δ es el tamaño de efecto a testear
X_i es el valor del voto de cada estudio
$\tilde{n} = (n^E+n^C) / (n^E*n^C)$ donde n's son las cantidades de sujetos experimentales de cada estudio **(3)**

Para mayores detalles de cómo aplicar las formulas indicadas remitirse a [11].

2.2.2 Response Ratio

Es el método de agregación recomendado dentro del ámbito de la ecología [15], El mismo consiste en estimar un índice de efecto, o Ratio, entre un tratamiento Experimental y otro de Control mediante el cociente de ambas medias ($RR = Y^E / Y^C$). Este cociente permite estimar la proporción de mejora existente entre ambos tratamientos. Así, por ejemplo, un ratio de 1.3 indicará que el tratamiento experimental es un 30% mejor que el de control, o un ratio de 1 indicará que no hay diferencias en el desempeño de ambos tratamientos.

Para que la combinación de un conjunto de estudios sea más precisa se le incorporó al método el logaritmo natural, que permite linealizar los resultados y normalizar su distribución, convirtiéndolo en un método apropiado para estimaciones de efectos cuando el conjunto de experimentos es pequeño [16]. Es importante destacar que una vez estimados el índice, se debe aplicar el anti-logaritmo al resultado para obtener el índice de efecto final. La aplicación del método consta de dos pasos, primeramente se debe estimar el Ratio de cada uno de los experimentos y luego, en base a éstos, se estima el Ratio o efecto global mediante un promedio ponderado de los ratios individuales, como se muestra en la función 4:

$$L^* = \frac{\sum_{i=1}^{k} W_i^* L_i}{\sum_{i=1}^{k} W_i^*}$$

L^* es el efecto global
L_i es el efecto de cada estudio
W_i es el factor de peso = $1/v$ **(4)**

Es importante destacar que el factor de peso de los estudios puede estimarse aplicando un método del tipo *paramétrico* (RRP) donde el ponderador de los estudios es la inversa de la varianza (como lo indica la función 5) o *no paramétrico* (RRNP), donde el ponderador de los estudios es la cantidad de sujetos experimentales (como lo indica la función 6). Siendo la principal ventaja del método no paramétrico el hecho de que no requiere conocer las varianzas de los estudios y la principal ventaja del método paramétrico su alta precisión aún cuando los estudios incluyen pocos sujetos.

$$v = \frac{S^{2E}}{n^E Y^E} + \frac{S^{2C}}{n^C Y^C}$$

v es el error típico
S^2‘s son las varianzas de los estudios
Y‘s son las medias de los estudios
n‘s son las cantidades de sujetos **(5)**

$$v = \frac{n_C + n_E}{n_E n_C} + \frac{Ln(RR^2)}{2(n_C + n_E)}$$

v es el error típico
n‘s son las cantidades de sujetos
RR es el Ratio **(6)**

Para mayores detalles de cómo aplicar las formulas indicadas remitirse a [15].

3. Aplicación de los Métodos de Agregación

3.1. El problema de la Agregación en la IS

Como se mencionó anteriormente, en la actualidad muchos investigadores han intentado aplicar MA en IS, pero a excepción de [7] que logró agregar 15

experimentos vinculados al rendimiento de los programadores cuando se trabaja de a pares, la mayoría de los trabajos no pudieron aplicarlo. Por ejemplo: en [4] se identificó un conjunto de experimentos vinculados a técnicas de prueba de software, pero no se pudo desarrollar el MA debido a la gran diversidad de técnica encontradas, a la falta de estandarización de variables respuesta y la falta de publicación de las varianzas en los informes; en [17] se identificó un conjunto de estudios experimentales que analizan el comportamiento de los métodos de estimación de software, pero no se pudo desarrollar el MA debido a la gran discrepancia entre los métodos identificados y las variables respuesta utilizadas; en [18] se analizó cuáles son los factores que motivan la adopción de un proceso de mejora basado en CMMI, pero no se pudo desarrollar el MA debido a la falta de estandarización de las variables respuesta y la baja calidad de los informes; en [19] se analizó el desempeño de los métodos de estimación generados a través de la experiencia recabada en una única compañía respecto de los métodos desarrollado en base a la experiencia recabada en varias compañías, pero no se pudo desarrollar el MA por un problema de compatibilidad entre los estudios identificados y la baja calidad de los informes; en [20] se analizó la reutilización del software en la modificación y/o creación de nuevos productos, pero no se pudo desarrollar el MA debido a la gran diversidad en las variables respuesta analizadas en cada estudio.
En resumen, podemos decir que los principales problemas que hoy afronta la comunidad científica en IS cuando decide realizar un MA son: la escasez de experimentos y/o replicaciones, discrepancias en las variables respuesta y la falta de estándares para informes de experimentos.

3.2. Desarrollo de la solución

Si bien existen tres problemas básicos a la hora de desarrollar un MA en IS, el problema de escasez de experimentos no puede ser solucionado por los métodos de agregación, pero contar con métodos de agregación más flexibles puede ayudar a reducir la cantidad de estudios descartados y aumentar así su número. En tal sentido, podemos decir que el RRNP no requiere conocer las varianzas para poder ser aplicado, lo que permita paliar en parte el problema de la baja calidad de los informes experimentales. El método CVE permite combinar los resultados de los experimentos aún cuando las variables respuestas utilizadas en cada experimento no sean las mismas, además no requiere conocer las varianzas ni las medias de los tratamientos. Ahora bien no todos los métodos de agregación presentan resultados con el mismo nivel de fiabilidad y precisión. Según el trabajo de [21] el RRP puede considerarse un método alternativo al DMP para su uso en medicina. En general, el método DMP tiene mayor precisión cuando se agregan muchos experimentos mientras que el RRP tiene mayor precisión cuando se agregan pocos experimentos [16]. El RRNP por su parte tiene un aceptable nivel de error independientemente de la cantidad de estudios que se agreguen pero su potencia es bastante limitada [16], ya que por su esencia no paramétrica requiere que las diferencias entre los tratamientos sean amplias para indicar que la misma es significativa, y el VCE es un interesante método de agregación cuando la cantidad de estudios a favor de cada uno de los tratamientos es similar [11], aumentando su precisión con el incremento de la cantidad de

experimentos [16]. A continuación se presentan un conjunto de pautas para poder aplicar los métodos de agregación presentados de forma conjunta.

3.3. Pautas para el uso de los métodos de agregación

Para determinar cuándo usar un método de MA u otro, se deben analizar sus restricciones de uso (parámetros estadísticos, relación entre las variables respuesta, etc.) y la cantidad de estudios identificados como base para determinar qué método utilizar en caso que pueda aplicarse más de uno. En tal sentido, podemos decir que si se cuenta con un conjunto de 15 o mas experimentos que publican medias, varianza, y cantidad de sujetos experimentales se recomienda utilizar DMP, por ser éste el método mas eficiente [16], pero, en caso de tener menos de 15 experimentos con estas características, se deberían agregar mediante el método RRP [16].
Por otra parte cuando los estudios presentan problemas de informes, se recomienda utilizar el RRNP si los estudios publican solo las medias y la cantidad de sujetos experimentales, y VCE si en los estudios en lugar de detallar las medias se limitan a indicar que un tratamiento es mejor que el otro. En la tabla 1 se describen los criterios antes mencionados.

El informes publica / Cantidad de estudios	Medias, varianzas y cantidad de sujetos experimentales	Medias y cantidad de sujetos experimentales	Diferencias entre Medias y cantidad de sujetos experimentales
>= 15	Usar DMP	Usar RRNP	Usar VCE
< 15	Usar RRP		

Tabla 1. Condiciones de aplicación de los métodos.

Se debe tener en cuenta que las condiciones descriptas en la tabla 1, solo se aplican si las variables respuesta publicadas por los experimentos unicamente difieren en la escala de valores utilizada, para el caso de variables respuesta no iguales el único método de agregación aplicable es VCE.
Para aumentar el nivel de evidencia de las conclusiones, se recomienda no utilizar los métodos de forma aislada, sino, por el contrario utilizarlos de forma conjunta, siempre y cuando el uso de alguno de ellos permita incrementar la cantidad de estudios factibles de ser agregados. Así, por ejemplo, si se cuenta con 5 experimentos agregables por RRP y tres por RRNP, la recomendación es agregar los primeros 5 con RRP y los 8 totales por RRNP, de esta forma se podrá generar distintos grupos de agregación que permitan ver como evoluciona el fenómeno con el aumento de la evidencia.
Es de hacer notar que cuando se utiliza más de un método de agregación en forma simultánea, los resultados obtenidos pueden no ser compatibles. Cuando esto sucede, se recomienda hacer un análisis para tratar de identificar variables no controladas que afecten al resultado obtenido por cada método, de forma similar a como se hace con el análisis de heterogeneidad.

4. Caso de Estudio

4.1 Presentación del caso de estudio

Para mostrar cómo aplicar los métodos de agregación bajo las pautas definidas, se tomó como base los estudios identificados en la RS desarrollado por [3], ya que los estudios están disponibles, y se contrastarán las conclusiones publicadas, en base a dicha RS, en el artículo [22]. Motiva esta elección el hecho de que en esta RS se ha identificado una gran variedad de técnicas de educción de requisitos, tanto tradicionales (como son las Entrevistas o Análisis de Protocolo) como alternativas (como son Card Sort y Ladderin grid entre otras), lo que permite realizar varios MA.

4.2 Comparación de Resultados

Para que el contraste de resultados con los métodos de agregación sea más claro, los resultados obtenidos se muestran mediante los conocidos diagramas de árboles. Es de hacer notar que dichos diagramas en lugar de estar dibujados de forma vertical, como habitualmente lo dibujan los programas especializados en MA, han sido dibujados de forma horizontal mediante Ms EXCEL debido a que no se ha podido identificar ningún programa que aplique los métodos de agregación alternativos. Tener en cuenta que los extremos de las líneas vinculadas al eje Y representan los límites del intervalo de confianza, el punto identifica el tamaño de efecto y el peso del estudio se representa mediante el valor que acompaña el código de experimento debajo del eje X.

4.2.1 Caso 1: El Análisis de Protocolo no aporta más conocimientos que la Entrevista

Esta conclusión fue obtenida en base a cuatro experimentos (E-1 [23], E-2 [24], E-3 [25] y E-4 [25]) que comparan Entrevista vs. Análisis de Protocolo (AP) analizando la variable ganancia (cantidad de cláusulas obtenidas). Dado que solo uno de estos estudios publica todos los parámetros estadísticos y los tres restantes publican medias y cantidad de sujetos experimentales, y no existen problemas con las variables respuesta, los mismos pueden agregarse mediante RRNP.

Como se aprecia en la figura 1, el tamaño de efecto global es apróximadamente 1,5, lo que implicaría que la entrevista es casi un 50 % mejor que el AP, pero, como el intervalo de confianza contiene al valor 1, la evidencia no puede ser considerada significativa al 95%. Por lo tanto, *se reafirma la no existencia de evidencia de que el AP es mejor que la entrevista.*

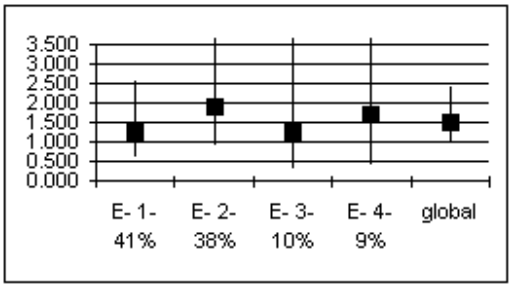


Fig. 1. Entrevista vs. AP (Ganancia).

4.2.2 Caso 2: Las Entrevistas insumen más tiempos que el Análisis de Protocolo
Esta conclusión fue obtenida en base a cuatro experimentos (E-1 [23], E-2 [24], E-3 [25] y E-4 [25]) que comparan Entrevista vs. Análisis de Protocolo (AP) analizando la variable esfuerzo (tiempo necesario para desarrollar la sesión y analizar los datos). Dado que solo uno de estos estudios publica todos los parámetros estadísticos y los tres restantes publican medias y cantidad de sujetos experimentales, y no existen problemas con las variables respuesta, los mismos pueden agregarse mediante RRNP.
Como se aprecia en la figura 2, el tamaño de efecto global es apróximadamente 1, lo que implicaría que no se espera que existan diferencias en el desempeño de ambas técnicas. Por lo tanto, *se contradice la afirmación original, no hay evidencias para afirmar que la entrevista consume más tiempo que el AP.*

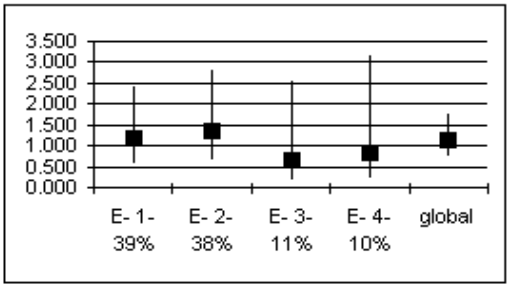


Fig. 2. Entrevista vs. AP (Esfuerzo).

4.2.3 Caso 3: Laddering Grid aporta más conocimiento que Card Sort
Esta conclusión fue obtenida en base a cuatro experimentos (E-1 [23], E-2 [24], E-3 [25] y E-4 [25]) que comparan las técnicas Laddering Grid (LD) vs. Card Sort (CS) analizando la variable ganancia (cantidad de cláusulas obtenidas). Dado que solo uno de estos estudios publica todos los parámetros estadísticos y los tres restantes publican medias y cantidad de sujetos experimentales, y no existen problemas con las variables respuesta, los mismos pueden agregarse mediante RRNP.
Como se aprecia en la figura 3, el tamaño de efecto global es apróximadamente 2, lo que implicaría que LG duplica el rendimiento de CS, mostrando además evidencias

significativas al 95%. Por lo tanto, *se reafirma el mejor desempeño de LG y se destaca que las diferencias son significativas*

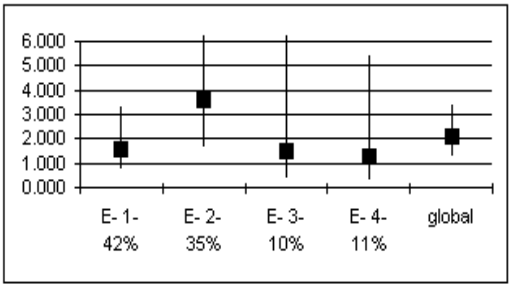


Fig. 3. LD vs. CS (Ganancia).

4.2.4 Caso 4: Laddering Grid consume más tiempo que Card Sort

Esta conclusión fue obtenida en base a cuatro experimentos (E-1 [23], E-2 [24], E-3 [25] y E-4 [25]) que comparan las técnicas Laddering Grid (LD) vs. Card Sort (CS) analizando la variable esfuerzo (tiempo necesario para desarrollar la sesión y analizar los datos). Dado que solo uno de estos estudios publica todos los parámetros estadísticos y los tres restantes publican medias y cantidad de sujetos experimentales, y no existen problemas con las variables respuesta, los mismos pueden agregarse mediante RRNP.

Como se aprecia en la figura 4, el tamaño de efecto global es apróximadamente 1,3, lo que implicaría que LG mejora un 30 % el desempeño de CS, pero la evidencia no es significativa al 95%, dado que el intervalo de confianza contiene claramente al valor 1. Por lo tanto, *se contradice la afirmación original, no hay evidencias para afirmar que la LG consume más tiempo que el CS.*

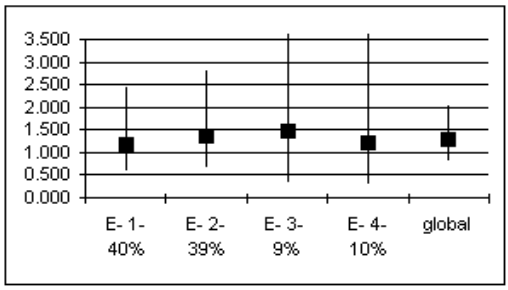


Fig. 4. LD vs. CS (esfuerzo).

4.2.5 Caso 5: La entrevista es más eficiente que CS

Esta conclusión fue obtenida en base a cinco experimentos (E-1 [23], E-2 [24], E-3 [25], E-4 [25] y E-5 [26]) que comparan las Entrevistas vs. Card Sort (CS) analizando la variable ganancia estimada como: la cantidad de cláusulas, en los primeros 4 experimentos, y la cantidad de reglas en el último. Dado que solo uno de estos estudios publica todos los parámetros estadísticos y los cuatro restantes publican medias y cantidad de sujetos experimentales, se realizarán dos agregaciones, la

primera incluirá cuatro experimentos mediante RRNP y la segunda incluirá a los cinco experimentos mediante VCE.
Como se aprecia en las figuras 5 y 6, las Entrevistas muestran un mejor comportamiento que CS. En la primer agregación el tamaño de efecto global es apróximadamente 1,35, lo que implicaría que la mejora alcanza el 35%, pero no es significativa al 95% da que el intervalo de confianza contiene al valor 1, en cambio en la segunda agregación, que incorpora un nuevo experimento, muestra resultados significativos y un tamaño de efecto medio similar al estimado anteriormente. Por lo tanto, *se reafirma el mejor desempeño de la entrevista y se destaca que las diferencias son significativas.*

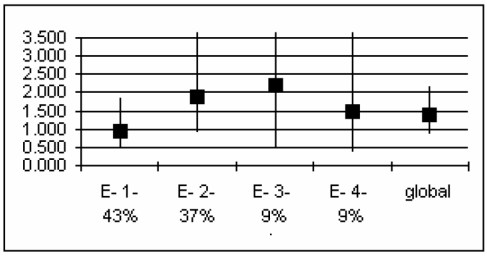


Fig. 5. Entrevista vs. CS (Ganancia) - RRNP

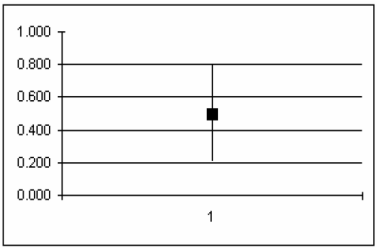


Fig. 6. Entrevista vs. CS (Ganancia) - VCE.

5. Discusión

Como se puede apreciar en los resultados obtenidos, en general, cuando todos los estudios apoyan a un mismo tratamiento, el resultado obtenido en el MA estadístico es compatible con el resultado original mediante CVNE, pero esto no puede tomarse como una regla universal, ya que en el caso de la combinación de los estudios vinculados a la Ganancia de LD vs. CS, a pesar de que todos los estudios apoyaban a la técnica LG, el resultado final no fue significativo. Esto refuerza la importancia de utilizar métodos estadísticos cuando se realiza un MA para evitar caer en errores de interpretación de los resultados.
También es importante el uso de MA cuando no existe un claro ganador entre los tratamientos, ya que en muchos casos pueden existir diferencias significativas a pesar de que no se advierta un claro ganador mediante el CVNE.
Por otra parte, el hecho de haber desarrollado dos agregaciones en la comparación de las Entrevistas vs Card Sort permitió, por un lado, afirmar que las diferencias son significativas, gracias al resultado del VCE, y por otro lado, ver como es la influencia de cada estudio en la conclusión final, gracias al RRNP.

6. Conclusiones

Se considera que la incorporación de los métodos de agregación RRNP y VCE ha sido altamente positiva. Si bien la mayoría de las nuevas conclusiones son compatibles con las tomadas originalmente en el artículo [22], el hecho de poder estimar un tamaño de efecto junto con su intervalo de confianza con una fiabilidad del 95% permite a los investigadores expresar sus conclusiones con mayor seguridad. Además mostrar los resultados a través de gráficas como el diagrama de árboles permite hacer un chequeo de heterogeneidad, un aspecto más que relevante a la hora de generar un resultado mediante la agregación de experimentos.
Dado que los métodos de agregación utilizadas han sido desarrolladas por investigadores de otras ramas de la ciencia, como son la medicina y la ecología, se considera de utilidad desarrollar nuevos trabajos tendientes a mostrar como es el comportamiento de estos métodos en un contexto experimental como el que hoy presenta la IS.

7. Financiamiento

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología del Gobierno de España, en el marco del proyecto TIN2008-00555.

8. Referencias

1. Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sörumgård, S., Zelkowitz, M.; 1996; *The empirical investigation of perspective-based reading*, International Journal on Empirical Software Engineering, Vol. 1, No. 2; pp. 133–164
2. Kitchenham, B. A.; 2004; *Procedures for performing systematic reviews*. Keele University; TR/SE-0401. Keele University Technical Report.
3. Davis, A.; Dieste o.; Hickey, A.; Juristo, N.; Moreno, A.; 2006; *Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review*; 14th IEEE Int. Requirements Engineering Conference (RE'06); pp. 179-188
4. Juristo N., Moreno A., Vegas S.; 2004; *Towards building a solid empirical body of knowledge in testing techniques*. ACM SIGSOFT Software Engineering Notes (SIGSOFT) 29(5); pp. 1-4
5. Jedlitschka, A. and Ciolkowski, M.; 2006; *Reporting Experiments in Software Engineering*; Franhofer Institute for Experimental Software Engineering.
6. Dyba, T., Kampenes, V., & Sjoberg, D.; 2006; *A systematic review of statistical power in software engineering experiments*. Information and Software Technology, 48(8); pp. 745-755.
7. Dyba, T., Aricholm, E.; Sjoberg, D.; Hannay J.; Shull, F.; 2007; *Are two heads better than one? On the effectiveness of pair programming*. IEEE Software; pp. 12-15.
8. Miller, J.; 2000; *Applying Meta-analytical Procedures to Software Engineering Experiments*. Journal of Systems and Software. (54): 1; pp. 29-39.
9. Glass, G; 1976; *Primary, secondary, and meta-analysis of research.* Educational Researcher 5; pp. 3-8

10. Goodman C.; 1996*; Literature Searching and Evidence Interpretation for Assessing Health Care Practices;* SBU; Stockholm.
11. Hedges, L.; Olkin, I.; 1985; *Statistical methods for meta-analysis*. Academic Press.
12. Mohagheghi, P., & Conradi , R.; 2004; *Vote-Counting for Combining Quantitative Evidence from Empirical Studies - An Example*. Proceedings of the International Symposium on Empirical Software Engineering (ISESE'04) .
13. Borenstein, M.; Hedges, L; Rothstein, H.; 2007; *Meta-Analysis Fixed Effect vs. random effect*; www.Meta-Analysis.com.
14. Sjoberg, D.; 2005; *A survey of controlled Experiments in Software Engineering*; IEEE Transactions on Software Engineering; Vol 31 Nro. 9
15. Gurevitch, J. and Hedges, L.; 2001; *Meta-analysis: Combining results of independent experiments.* Design and Analysis of Ecological Experiments (eds S.M. Scheiner and J. Gurevitch); pp. 347–369. Oxford University Press, Oxford.
16. Lajeunesse, M & Forbes, M.; 2003; *Variable reporting and quantitative reviews: a comparison of three meta-analytical techniques*. Ecology Letters, 6; pp. 448-454.
17. Jřrgensen, M.; 2004; A *Review of Studies on Expert Estimation of Software Development Effort*. Journal of Systems and Software. (70): 1-2; pp. 37-60.
18. Staples, M; Niazi,M; Ross Jeffery, D; Abrahams, A; Byatt, P; Murphy, R; 2007: *An exploratory study of why organizations do not adopt CMMI*. Journal of Systems and Software 80(6); pp. 883-895
19. Kitchenham, B.; 2007; *Cross versus Within-Company Cost Estimation Studies: A Systematic Review*; IEEE Transactions on Software Engineering; Vol 33 Nro. 5.
20. Mohagheghi, P., & Conradi , R.; 2004; *Vote-Counting for Combining Quantitative Evidence from Empirical Studies - An Example*. Proceedings of the International Symposium on Empirical Software Engineering (ISESE'04) .
21. Friedrich, J, Adhikari, N; Beyene, J; 2008; *The ratio of means method as an alternative to mean differences for analyzing continuous outcome variables in meta-analysis: A simulation study*; BMC Medical Research Methodology
22. Dieste, O.; Juristo, N.; Shull, F.; *Understanding the Customer: What Do We Know about Requirements Elicitation? Software*, IEEE; Volume 25, Issue 2, March-April 2008; pp. 11 – 13.
23. Burton, A., Shadbolt, N., Hedgecock, A. y Rugg, G.; 1988; *A Formal Evaluation of Knowledge Elicitation Techniques for Expert Systems: Domain 1*. Proceedings of Expert Systems '87 on Research and Development in Expert Systems IV**.** Pág. 136-145.
24. Corbridge, C., Rugg, G., Major, P., Shadbolt, N. y Burton, A. 1994. *Laddering: Technical and Tool in Knowledge Acquisition*. Department of Psychology, University of Nottingham.
25. Burton, A., Shadbolt, N., Rugg, G. y Hedgecock, A.; 1990. *The Efficacy of Knowledge Elicitation Techniques: A Comparison Across Domains and Level of Expertise*. Knowledge Acquisition 2(2): 167-178.
26. Schweickert, R., Burton, A., Taylor, N., Corlett, E., Shadbolt, N., Rugg, G. y Hedgecock, A.; 1987. *Comparing Knowledge Elicitation Techniques: A Case Study*. Artificial Intelligence Review (1): 245-253.

Adapting Software by Identifying Volatile and Aspectual Requirements

José M. Conejero[1], Juan Hernández[1], Ana Moreira[2] and João Araújo[2]

[1] Quercus Software Engineering Group, University of Extremadura, Spain
{chemacm , juanher}@unex.es
[2]CITI/Dept. Informática, FCT, Universidade Nova de Lisboa, Portugal
{ja , amm}@di.fct.unl.pt

Abstract. The software market demands systems with highly volatile requirements. Thus, time and effort to adapt software to business's changes must be reduced as much as possible. However, such volatile requirements are usually tangled with other concerns making systems evolution and adaptation difficult. In this paper, we propose a conceptual framework to identify both crosscutting and volatile requirements. This framework is based on a crosscutting pattern and uses traceability matrices and matrix operations. We also show an aspect-oriented refactoring process to externalize such requirements enhancing the future management of changes.

Keywords: Software adaptation, Requirements, Aspect-orientation, Crosscutting, Volatility.

1 Introduction

Enhancing business performance requires systems to be able to cope efficiently with changes in requirements, thus reducing time and development effort for adapting software. A key barrier to the success of these systems is the impact of requirements volatility on the system. As Firesmith [7] says: *"The more volatile the requirements, the more important it becomes for the requirements process to support the quick and easy modification and addition of requirements"*. However, such requirements are usually tangled with other concerns or requirements making evolution and adaptation of systems difficult and restricting maintainability of software.

In order to improve software adaptability, in [10] the authors present an approach to cope with those volatile concerns by externalizing them. They propose the use of aspect-oriented techniques to isolate such concerns in the same way as crosscutting concerns. However, the identification of volatile (usually tangled with other concerns) and crosscutting concerns is not trivial and it is usually left to the developer's expertise. In that sense, in [5] we proposed a conceptual framework where crosscutting is formally defined by means of a crosscutting pattern. This formal definition allows the distinction between the terms of crosscutting and tangling or scattering (often used without any difference).

In this paper, we apply the conceptual framework presented in [5] for the identification of not only crosscutting concerns but tangled concerns (where volatile

concerns are usually involved) as well. Then, the main contributions of the paper, also with respect to our previous work are twofold: first, the paper shows the applicability of the framework to cope with situations where adaptability and evolution must be improved. Since volatile concerns are modularized using aspect-oriented techniques, the system may be evolved just composing different models (model instantiation and composition). Second, the utilization of our conceptual framework helps in automating the process of identifying concerns which must be isolated, allowing an aspectual refactoring process using the model notation introduced in [10]. According to [17], we perform a functional and technical software adaptation, depending on whether it is adapted for handling with volatile (new or changed user needs) or crosscutting concerns respectively.

The rest of paper is organized as follows: Section 2 introduces some background on an aspect-oriented approach for coping with volatility [10] and the conceptual framework presented in [5]. In Section 3 we show the application of the conceptual framework used to identify both crosscutting and volatile concerns at requirements level (illustrated by a case study). We also show in this section how to isolate such concerns using Use Case Pattern Specification. Finally, Section 4 presents related works and Section 5 draw some conclusions.

2 Background

In this section we introduce the approach presented in [10] and the crosscutting pattern and the matrix operations [5] used in this paper to improve the identification of tangled (which usually include volatile concerns) and crosscutting concerns.

2.1 AORE: Handling Volatility

Volatile requirements are business rules with a high degree of evolution. Such requirements may be changed quickly at any time. In [10], the authors propose an approach to cope with volatile concerns by separating or externalizing them (using aspect-oriented techniques).

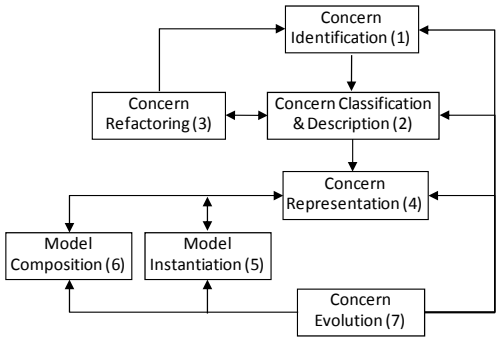


Fig. 1. Aspect-oriented evolutionary model for volatile concerns (taken from [10])

Once the concerns have been modularized they can be composed with the base system later on. The approach focuses on requirements evolution, where classification, composition and instantiation are the most important tasks to achieve a highly evolvable system. The approach consists of several steps (Fig. 1 shows the process outline):

1) The main concerns of the problem domain are identified. A concern refers to a matter of interest that the systems must address to satisfy the stakeholders needs.
2) The concerns identified are classified into several groups: either service or constraint, and either enduring or volatile. This classification leads to a description of each concern.
3) After classifying and describing concerns, the authors externalize the concerns using aspect-oriented techniques.
4) Concerns are then represented using UML use cases, activity models and pattern specifications [8]. Crosscutting and volatile concerns are represented and defined as roles elements in the representation models.
5) The volatile concerns are then instantiated by means of simple rules. This instantiation is needed to perform the next composition step.
6) Finally, the concerns isolated can be weaved into a base model.
7) The steps from 1 to 5 can be iteratively repeated to model new requirements allowing concern evolution.

2.2 The crosscutting pattern

In [5] we presented a conceptual framework where crosscutting can be clearly distinguished from scattering and tangling, allowing to formally define these terms. These definitions are based on the study of trace dependencies through an extension to traceability matrices. In particular, our proposition is that crosscutting can be defined in terms of a mapping relationship property. From a mathematical point of view this means that we have two domains (called Source and Target) related to each other through a mapping or traceability relationship. We use the term of Crosscutting Pattern to denote this situation (see Fig. 2).

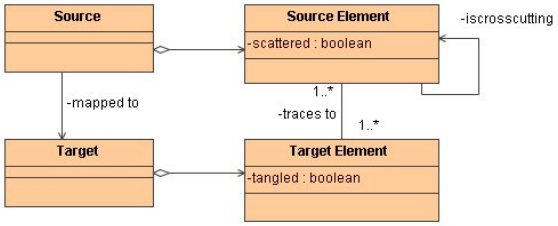


Fig. 2. The crosscutting pattern

The terms scattering, tangling and crosscutting are defined as specific cases of the mapping between “source” and “target”. We say that scattering occurs when, in a mapping between source and target, a source element is related to multiple target elements. Similarly, we can focus on the relation between target elements and source elements. This relation (here also called mapping) is the reverse of the previous one. So, we say that tangling occurs when, in a mapping between source and target, a

target element is related to multiple source elements. There is a specific combination of tangling and scattering which we call crosscutting, defined as follows: Crosscutting occurs when, in a mapping between source and target, a source element is scattered over target elements and where in at least one of these target elements, some other source element is tangled.

We use the dependency matrix (a special kind of traceability matrix) to represent the mappings between source and target. From this dependency matrix, we derive two other ones called scattering and tangling matrices, which show scattered and tangled concerns of a system, respectively. Performing simple matrix operations (product of scattering and transpose of tangling), a new matrix (called crosscutting product matrix) is obtained which shows the frequency of crosscutting relationships (by means of concern metrics). We derive the final crosscutting matrix just converting the crosscutting product matrix into a binary matrix. The crosscutting matrix provides the crosscutting concerns of a system (see Fig. 3). See [5] to obtain more details.

This approach can be applied to the early phases of the software development, handling concerns and requirements represented by use cases (as source and target respectively), but also to the later development phases, handling UML designs and Java code, for example. In this paper we focus on the early phases, and therefore study the viability of the framework to identify volatile and crosscutting concerns at requirements level.

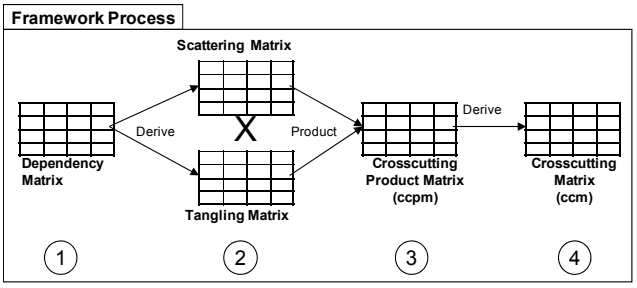


Fig. 3. Overview of steps in the framework

3. Externalizing volatile and crosscutting concerns in requirements

In this section we describe the main steps we must perform to distinguish between concerns involved in tangling (usually volatile concerns) and concerns involved in crosscutting. The section consists of several tasks which drives the identification. First of all we must identify the concerns in the system and model the requirements using some language. Secondly we apply the crosscutting pattern and the concepts defined in it to classify the concerns. Finally, we externalize the volatile and aspectual concerns using aspect-oriented techniques.

3.1 Concern Identification (1)

In order to identify the concerns of the system (step (1) in Fig. 1), we follow the tasks suggested in [10], i.e, identification of stakeholders, inspection of documents that

describe the problem, existing catalogs [4], stakeholders' interviews transcripts and searching techniques [14]. In order to illustrate the approach, we show an example where we apply the different steps. The example is inspired on the functionalities offered by the Washington subway system:

"To use the subway, a client uses a card that must have been credited with some amount of money. A card is bought and credited in buying machines available in subway stations. The card is used in an entering machine to initiate a trip. When the destination is reached, the card is used in an exit machine that debits it with an amount to be paid. If the card has not enough credit the gate will not open"

We identified Client and Passenger as final users of the system (a client may be a potential passenger). Based on the description of the system, the requirements which the system must meet are represented in Table 1. On the other hand, we identified the key concerns represented in Table 2 (the process used to identify requirements and concerns is out of the scope of this paper).

Table 1. Requirements of the system

Req.	Description
R1	A client buys a card in a buying machine
R2	A client must own a valid card
R3	Clients credit cards with minimum amounts of money in buying machines
R4	A client enters a subway station using a card in an entry machine
R5	A client leaves the subway station using his card in an exit machine that debits it with the cost of the trip. If the card has not enough credit the gate will not open
R6	The system is used for several passengers simultaneously
R7	The system needs to react in time to avoid delaying passengers while they are entering or leaving the subway, or crediting their cards
R8	The system must be available for use

As we can see in these tables, we have added some requirements and concerns regarding to some quality attributes or constraints under which the system must operate (MultiAccess, Response time and Availability).

Table 2. Concerns in the subway system

Concern	Concern name	Concern	Concern name
C1	Buy card	C7	Calculate fare
C2	Validate card	C8	Condition to leave
C3	Credit card	C9	MultiAccess
C4	Minimum amount	C10	Response time
C5	Enter subway	C11	Availability
C6	Exit subway		

3.2. Modeling requirements

Once we have identified the concerns of the system, we model the requirements of the system using some requirements modeling language or notation. In our case, we use UML use case diagrams [11] for modeling requirements. The purpose of modeling requirements is to have a first representation of the functionality which the system must implement. In Fig. 4 we show a simple use case diagram for the Subway system.

We know now the main functionalities that the system must perform (concerns) and we have a first model for these functionalities (use cases). Taking these concerns and use cases as source and target domains respectively, in the next step we build a dependency matrix where mappings between concerns and use cases are represented.

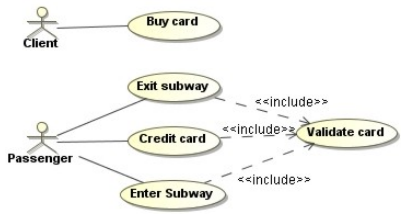


Fig. 4. Use case diagram for the Subway system

3.3 Classifying concerns (2)

In [10] the authors divide the concerns into four different categories depending on two different factors. Depending on longevity, concerns can be enduring or volatile and depending on nature they can be services or constraints. Enduring concerns are "stable requirements which derive from the core activity of the organization". Volatile concerns are "likely to change during the system development or after the system has been put into operation". On the other hand, constraints are properties that the system must satisfy (related to non-functional requirements) whereas services define functionality that the system must offer (related to functional requirements). Since we are interested in adaptation and evolution, we focus on a different classification dividing concerns into three categories: isolated (in an ideal system, every functional requirement should be isolated), tangled (this category often includes volatile concerns) and crosscutting (constraints are usually included in this category).

Table 3. Dependency matrix (concerns x use cases) for the subway system

		Use cases					
		Buy card	CreditCard	ValidateCard	ExitSubway	EnterSubway	
Concerns	Buy card	1	0	0	0	0	NS
	Validate card	0	1	1	1	1	S
	Credit card	0	1	0	0	0	NS
	Minimum amount	0	1	0	0	0	NS
	Enter subway	0	0	0	0	1	NS
	Exit subway	0	0	0	1	0	NS
	Calculate fare	0	0	0	1	0	NS
	Condition to leave	0	0	0	1	0	NS
	MultiAccess	1	1	0	1	1	S
	Response time	0	0	1	1	1	S
	Availability	1	1	0	1	1	S
		T	T	T	T	T	

In order to classify the different concerns of the system, we apply the matrix operations defined in our conceptual framework (see [5]). By these simple matrix operations, tangled and crosscutting concerns are identified. Firstly, we build a dependency matrix where we show concerns (in rows) and use cases (in columns) considered as source and target respectively. In Table 3 we can see the dependency matrix for the Subway example used in the paper. As we can, each concern is related to the use cases which contribute or address it. There are also some relations inferred

from the requirements of the system, e.g. when a passenger credits a card, the system must ensure that a minimum amount of money is introduced (requirement R3 in Section 3.1). This is the reason why the Minimun amount concern is mapped onto the CreditCard use case. Similarly, when a passenger leaves the subway, the system must charge cost of the trip and check that the card has enough money to pay the trip (requirement R5). Then, Calculate fare and Condition to leave concerns are related to the ExitSubway use case. As a different example almost all the use cases are including the ValidateCard use case (see Fig. 4). Then we assume that the Validate card concern is addressed by ValidateCard use case and also by all the use cases including it. Finally, the non-functional concerns (Availability, MultiAccess and Response time) are also related to the use cases constrained by these concerns.

Table 4. Scattering matrix (concerns x use cases) for dependency matrix of Table 3

		Use cases				
		BuyCard	CreditCard	ValidateCard	ExitSubway	EnterSubway
Concerns	Buy card	0	0	0	0	0
	Validate card	0	1	1	1	1
	Credit card	0	0	0	0	0
	Minimum amount	0	0	0	0	0
	Enter subway	0	0	0	0	0
	Exit subway	0	0	0	0	0
	Calculate fare	0	0	0	0	0
	Condition to leave	0	0	0	0	0
	MultiAccess	1	1	0	1	1
	Response time	0	0	1	1	1
	Availability	1	1	0	1	1

Taking the dependency matrix (Table 3), we derive two different matrices where we represent scattering and tangling of the system (Table 4 and Table 5 respectively).

Table 5. Tangling matrix (use cases x concerns) for dependency matrix of Table 3

		concerns										
		Buy card	Validate card	Credit card	Minimum amount	Enter subway	Exit subway	Calculate fare	Condition to leave	MultiAccess	Response time	Availability
Use cases	BuyCard	1	0	0	0	0	0	0	0	1	0	1
	CreditCard	0	1	1	1	0	0	0	0	1	0	1
	ValidateCard	0	1	0	0	0	0	0	0	0	1	0
	ExitSubway	0	1	0	0	0	1	1	1	1	1	1
	EnterSubway	0	1	0	0	1	0	0	0	1	1	1

By a simple product of scattering and tangling matrices, we obtain a new matrix called crosscutting product matrix (representing the frequency of crosscutting). Since we are not interested in frequency, we just derive the final crosscutting matrix from the crosscutting product matrix (see the process in [5]). The crosscutting matrix obtained for the Subway example is presented in *Table 6*.

Table 6. Crosscutting matrix (concerns x concerns) for the subway system

		concerns										
		Buy card	Validate card	Credit card	Minimum amount	Enter subway	Exit subway	Calculate fare	Condition to leave	MultiAccess	Response time	Availability
Concerns	Buy card	0	0	0	0	0	0	0	0	0	0	0
	Validate card	0	0	1	1	1	1	1	1	1	1	1
	Credit card	0	0	0	0	0	0	0	0	0	0	0
	Minimum amount	0	0	0	0	0	0	0	0	0	0	0
	Enter subway	0	0	0	0	0	0	0	0	0	0	0
	Exit subway	0	0	0	0	0	0	0	0	0	0	0
	Calculate fare	0	0	0	0	0	0	0	0	0	0	0
	Condition to leave	0	0	0	0	0	0	0	0	0	0	0
	MultiAccess	1	1	1	1	1	1	1	1	0	1	1
	Response time	0	1	0	0	1	1	1	1	1	0	1
	Availability	1	1	1	1	1	1	1	1	1	1	0

As we can see in the matrix, on one hand we obtain several crosscutting concerns: Validate card, MultiAccess, Response Time and Availability. We classify these concerns as crosscutting. These concerns are candidates to be externalized by means of aspect-oriented techniques. On the other hand, we are also interested in tangled concerns since volatile concerns are usually tangled with other services that the system provides. In order to identify such concerns, we focus on the tangling matrix (see Table 5). In this matrix we can observe use cases where functionality of several concerns is scrambled. In Table 7, we show the use cases and the concerns tangled in them.

Table 7. Use cases where several concerns are tangled.

Use Cases	**Concerns tangled in the use cases**
BuyCard	Buy card, MultiAccess, Availability.
CreditCard	Validate card, Credit card, Minimum amount, MultiAccess, Availability.
ValidateCard	Validate card, Response time.
ExitSubway	Validate card, Exit subway, Calculate fare, Condition to leave, MultiAccess, Response time, Availability.
EnterSubway	Validate Card, Enter subway, MultiAccess, Reponse time, Availability

We remove now from Table 7 the concerns identified as crosscutting in the crosscutting matrix (see Table *6*). The reason for this removal is clear: since these concerns have already been identified to be externalized we disregard them to obtain only tangled concerns. Table *8* shows the previous table without the crosscutting concerns (Validate card, MultiAccess, Response Time and Availability). In this table we can see that only CreditCard and ExitSubway use cases are tangled with functionality of several concerns (which they are not crosscutting concerns). These tangled concerns are the candidate to be classified as volatile. We focus first on CreditCard use case. In CreditCard there are two concerns involved, i.e. Credit card and Minimum amount. We consider Credit card a service. A client will always need to credit a valid card in order to enter the subway. However, we consider Minimum amount as a volatile concern since the Minimum amount to let the client credit his/her card may change (even several times in a short period of time).

Table 8. Use cases tangled without crosscutting concerns

Use Cases	Concerns tangled and not crosscutting
BuyCard	Buy card.
CreditCard	Credit card, Minimum amount.
ValidateCard	
ExitSubway	Exit subway, Calculate fare, Condition to leave.
EnterSubway	Enter subway.

In ExitSubway use case we can see three tangled concerns. Exit subway is considered as service (it will not change in time, the client will always leave the subway), whereas the other two concerns are considered volatile. The amount to be paid represents a behavior which may change in the future. For example, to calculate the cost of the trip there are several options, which can range from fixed prices to a price depending on number of zones traveled or to special discounts (if promotions are available). Finally, the Condition to leave is also considered as volatile since we may want in the future to allow some passengers travel for free (e.g. handicapped people or people who work for the company). Thus, the process decreases the time to market since it points out the situations where developers must focus on. In Table 9 we can see the concerns of the system classified according to their type.

Table 9. Concerns' classification

	Isolated	Crosscutting	Tangled	
			Services	Volatile
Concerns		C2,C9,C10,C11	C1,C3,C5,C6	C4,C7,C8

The main purpose of this classification is to select the concerns that are candidate to be externalized. The crosscutting concerns are candidate to be refactored. We consider also volatile to be isolated using the same techniques than for crosscutting (aspect-oriented). Then, we aim at improving the number of isolated concerns (0 in the current classification). In next sections, we show the separation of crosscutting and volatile concerns by aspect-oriented techniques.

3.4 Concern representation (4)

Each concern can be described now in a different template, where its structural and internal information is shown as well as its relationships with other concerns. In Table 10 we can observe two template examples for Exit subway and Calculate fare. To obtain more information about these templates, see [10].

Table 10. Templates for Exit subway and Calculate fare

Concern #	C6
Name	Exit subway
Cassification	Tangled (service)
Stakeholders	Passenger
Interrelationships	C2,C7–C11
List of pre-conditions	
(1) Card is valid	
List of responsibilities	
(1) Check balance	
(2) Debit card	
(3) Register trip	
(4) Open gate	
(5) Eject card	

Concern #	C7
Name	Calculate fare
Cassification	Tangled (volatile)
Stakeholders	
Interrelationships	C6,C8–C11
List of responsibilities	
(1) Get entry station	
(2) Get exit station	
(3) Calculate price	

3.5 Model Instantiation (5)

After classifying and representing the concerns, use cases where several concerns are tangled or crosscut are separated into several use cases. We use the Use Case Pattern Specification presented in [10] to model these changes. Pattern Specifications (PSs) [8] are a way of formalizing the reuse of models. The notation for PSs is based on UML [11]. A PS describes a pattern of structure or behavior defined over the roles which participants of the pattern play. Role names are preceded by a vertical bar ("|"). A PS can be instantiated by assigning concrete modeling elements to play these roles. A role is a specialization of a UML metaclass restricted by additional properties that any element fulfilling the role must possess. Hence, a role specifies a subset of the instances of the UML metaclass. A model conforms to a PS if its model elements that play the roles of the PS satisfy the properties defined by the roles. Thus, a conforming diagram must instantiate each of the roles with UML model elements, multiplicity and other constraints. Note that any number of additional model elements may be present in a conforming diagram as long as the role constraints are maintained. As in [16], we extend the notion of pattern specification from that of [8] by allowing both role elements and concrete modeling elements in a PS.

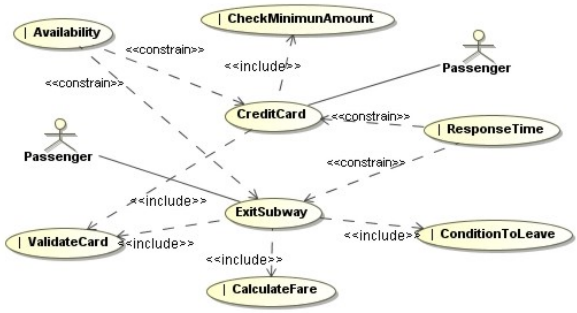


Fig. 5. Use Case Pattern Specification for subway system

In order to use this Pattern Specification, both crosscutting and volatile concerns are marked as roles (see "|" mark in Fig. 5). In addition, a new relationship is included to complement the typical <<include>> and <<extend>> ones. The new relationship is called <<constrain>> and is used by requirements which restrict the behaviour of other requirements. In Fig. 5 we can see part of the resulting model for the subway system (the whole diagram is not shown for space reasons). Note that the requirements regarding to crosscutting or volatile concerns have been modeled in a separate use case and maintain <<constrain>> or <<include>> relationships with the base use cases. Managing such requirements by aspect-oriented techniques allows the developer to obtain a high degree of evolution and adaptability in the systems build.

3.6 Model Composition (6)

The requirements engineer may quickly cope with business rules' changes without altering the rest of the system. For example, if the business rules to calculate the trip fare is based on a fixed price the use case role |*CalculateFare* can be instantiated with the following rule (see [10] to obtain more details):

```
Replace |CalculateFare
        with CalculateFareBasedOnFixedPrice
```

However, if the business rules changes for the trip fare to be based on zones, the use case role |*CalculateFare* can be instantiated with a rule of the type:

```
Replace |CalculateFare
        with CalculateFareBasedOnZones
```

Therefore, externalizing volatility promotes system evolution and adaptability, as changing the behavior of the system does not require to change what is already specified, but simply add new concerns and define new (or refine) existing instantiation rules. In [10], the authors also show how to perform a model composition to weave the isolated behavior with the base model (using composition rules and UML activity diagrams). The composition is not shown here since we focus on isolation of volatile and crosscutting behavior and not composition.

4. Related Work

In the area of Early Aspects, there are several authors who have presented methodologies for the identification of crosscutting concerns at requirements level, for example, the approach presented in [1] and [15]. However, this approach is focused only on requirements phases while the crosscutting pattern used here could be applied in any phase of the software life cycle [3][5].

Several authors have used matrices (design structure matrices, DSM) to analyze modularity in software design [2]. Lopes and Bajracharya [9] describe a method with clustering and partitioning of the design structure matrix for improving modularity of object-oriented designs. Even in [13], a relationship matrix (concerns x requirements) is described very similar to our dependency matrix, and used to identify crosscutting concerns. However, there is no formalized definition of crosscutting.

In [12] the authors propose the adaptation of software systems by means of the utilization of aspect-oriented techniques. They model the systems using UML and perform model checking to ensure that the system does not contain any inconsistence. While this work focuses on solving some interoperability problems after adapting software system using aspects, our work focuses on the identification of crosscutting and tangled behaviour in order to modularize it (by aspect-oriented techniques).

5. Conclusions and further work

We proposed a framework to identify volatile and crosscutting concerns at requirements level. This framework is based on a crosscutting pattern where definition of crosscutting is formalized by means of trace relationships. We visualize such relationships by means of traceability matrices. The identification of volatile concerns is based on tangling in the system since such concerns are usually jumbled with other concerns. Crosscutting concerns are identified by means of simple matrix operations. Once the volatile and crosscutting concerns have been identified, the requirements can be refactored so that each concern is modularized in a single entity. The UCPS representation illustrates the refactorization. This is particular useful in

model-driven software engineering where model transformations could take into account volatile concerns and the system may be built based on these transformations (our future work). As we have showed, our conceptual framework has different application areas in addition to identification of crosscutting concerns. In other publications we have shown other interesting applications such as traceability of crosscutting concerns throughout a whole development process [5], identification of crosscutting concerns at design level [3] or the definition of a concern-oriented metrics to assess modularity of systems [6].

References

1. Araujo, J., Moreira, A., Brito, I. & Rashid, A. (2002). *Aspect-Oriented Requirements with UML*, In Workshop on AOM with UML at International Conference on UML, Germany
2. Baldwin, C.Y. & Clark, K.B. (2000). *Design Rules vol I, The Power of Modularity*. MIT Press
3. Berg, K. van den, Conejero, J. & Hernández, J. (2006). *Identification of crosscutting in software design*. In Aspect-Oriented Modeling Workshop at 5th AOSD, Bonn
4. Chung, L., Nixon, B., Yu, E. & Mylopoulos, J. (2000) *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, 2000
5. Conejero, J. M., Berg, K. van den & Hernández, J. (2007). *Analysis of Crosscutting in Early Software Development Phases based on Traceability*. In Transactions on AOSD III, Volume 4620/2007, ISBN: 978-3-540-75161-8.
6. Conejero, J., Figueiredo, E., Garcia, A., Hernández, J. & Jurado, E. (2009) Early Crosscutting Metrics as Predictors of Software Instability. To appear in *47th International Conference Objects, Models, Components, Patterns (TOOLS Europe)*. Zurich, Switzerland.
7. Firesmith, D. G. (2004). *Creating a Project-Specific Requirements Engineering Process*, in Journal of Object Technology, vol. 3, no. 5, 2004, pp. 31-44.
8. France, R., Kim, D., Ghosh, S. & Song, E. (2004) *A UML-Based Pattern Specification Technique*, IEEE Transactions on Software Engineering, Volume 30(3), 2004
9. Lopes, C.V. & Bajracharya, S.K. (2005). *An analysis of modularity in aspect oriented design*. In 4th International AOSD Conference. Chicago, Illinois
10. Moreira, A., Araujo, J. & Whittle, J. (2006) *Modeling Volatile Concerns as Aspects*. In 18th CAISE. LNCS 4001/2006: 544-558. ISBN: 978-3-540-34652-4, Luxembourg.
11. OMG (2005). *UML Specification, version 2.0*, August 2005, http://www.omg.org
12. Pérez-Toledano, M., Navasa, A., Murillo, J.M. & Canal, C. (2007). *Safe Dynamic Adaptation using Aspect-Oriented Programming.* In 4th Workshop on Coordination and Adaptation Techniques for Software Entities, in conjunction with ECOOP'07, Germany.
13. Rashid, A., Moreira, A. & Araujo, J. (2003). *Modularisation and Composition of Aspectual Requirements.* In Second AOSD Conference. Boston, USA
14. Sampaio, A., Chitchyan, R., Rashid, A. & Rayson, P. (2005). *EA-Miner: A Tool for Automating Aspect-Oriented Requirements Identification*, ASE'05, IEEE Computer Society
15. Sampaio, A., Loughran, L., Rashid, A. & Rayson, P. (2005). *Mining Aspects in Requirements*. In Early Aspects 2005 Workshop at AOSD Conference. Chicago, USA
16. Whittle, J. and Araujo, J. (2004). *Scenario Modeling with Aspects*, in *IEE Proceedings Software*, Vol. 151, no. 04, pp. 157-172.
17. Yahiaoui, N., Traverson, B. & Levy, N. (2004), *Classification and comparison of Adaptable Platforms.* In 1st International Workshop on Coordination and Adaptation Techniques for Software Entities, in conjunction with ECOOP'04, Oslo, Norway.

A Tool-supported Natural Requirements Elicitation Technique for Pervasive Systems centred on End-users *

Francisca Pérez and Pedro Valderas

Centro de Investigación en Métodos de Producción de Software
Universidad Politécnica de Valencia
Camino de Vera, s/n, 46022 Valencia, Spain
{mperez, pvalderas}@pros.upv.es

Abstract. It is very important that end-users participate in the requirement elicitation process because they are the "owners" of the problem. In this paper, we present a natural requirements elicitation technique for pervasive systems centred on end-users that provide them a tool-supported natural visual language in order to describe the main characteristics of their pervasive system. Furthermore, we present a requirement elicitation process which guides end-users in the description.

1 Introduction

In this work, we present a tool-supported requirements elicitation technique for pervasive systems centred on end-users. End-users are the ones who have more in-depth knowledge about both the services that must be provided by the system and the environment in which the system is going to be deployed. Thus, we think it is very important that end-users actively participate with analysts in the description of the requirements of a pervasive system. However, end-users have no knowledge about requirements engineering techniques or computation in general. Thus, end-users need to be provided with techniques and tools that allow them to specify their requirements in a *natural* way. Natural is a concept used in the field of end-user development and it is defined as [1] "faithfully representing nature or life," which implies that it works in accordance with the way people expect. Thus, our main goal is to define a natural requirements elicitation technique that helps end-users to describe the system they need. To do this, we present a tool whose interface is inspired by well-accepted techniques and metaphors in the field of end-user development. This tool allows end-users to describe the environment in which the system must be deployed, the devices that are available, and the services that these devices must provide end-users with. Note that this tool constitutes a valuable mechanism to be used in requirements elicitation activities that helps analysts to identify the user needs that the system must satisfy. Finally, we have defined a requirements elicitation process that precisely determines the steps that end-users must follow to create a system description and how end-users and analysts interact to each other along this process.

* This work has been developed with the support of MEC under the project SESAMO TIN2007-62894 and cofinanced by FEDER, in the grants program FPI.

2 Supporting end-user with elicitation tools

In order to allow end-users to describe their needs, we have developed a prototype of an elicitation tool. To do this, we have studied first end-users population. Our study concludes that end-user population is quite diverse because end-users are present in a lot of domains and with different needs [2] and we have observed two main common profiles: (1) **End-users** who are not familiarized with computers applications and (2) **Advanced** end-users who computers applications are familiar for them but they are not a professional (e.g. doctors, accountant...). Our main goal is to provide these two profiles with our tool.

These profiles are supported by different **types of interfaces**: closed-option and open-option. End-users are supported by closed-option interfaces and advanced end-users are supported by both interfaces. Closed-option interfaces provide end-users with a **catalogue** of requirements. This catalogue allows end-users to select those requirements that satisfy their needs. In this case, the catalogue offers end-users pervasive requirements, which include the available services and a default configuration for them. Open-option interfaces allow advanced end-users to define new requirements if the requirements catalogue does not satisfy their needs. To support this, our tools need to offer a mapping between the information which represents the catalogue and the user's requirements with the computer's interface and technologies for it. To do this mapping, we use a visual language. To define this visual language and interfaces, we have inspired by Natural Programming [1], Visual Programming [3], Programming By Example [4] and Jigsaw metaphor [5]. In our tool, we have represented the catalogue using the Feature Modelling technique [6]. In order to tackle the representation of the catalogue in the proper interface, we use a Controller. The **Controller** also assists end-users in the description of their needs and it checks that the description of end-users are completed and without errors. The Controller is implemented using Eclipse Modelling Framework (EMF) and FAMA Framework [7]. Furthermore, a **repository** is used to save the end-user system description. Fig. 1 shows a smart example of the catalogue which supports the available services and devices for a smart home modelled as a feature model and the architecture which supports our tool.

Fig. 3 shows a snapshot of our prototypical tool. The name of the current view is represented at the top, a title which indicates what are required by the tool and the buttons to navigate among views. The available devices for the environment are represented on the left together with a representative image. On the right of the figure is represented the environment, their locations and the devices previously selected. Finally, at the bottom of the figure, the information area, where our tool can advice to end-users or assist them, is shown.

3 Requirement Elicitation Process

We have defined a requirements elicitation process in which end-users actively participate describing their needs by using the tool presented above. This process determines the steps that end-users must follow to create a system description and how end-users and analyst must interact with each other. Thus, we propose four phases (see Fig. 2):

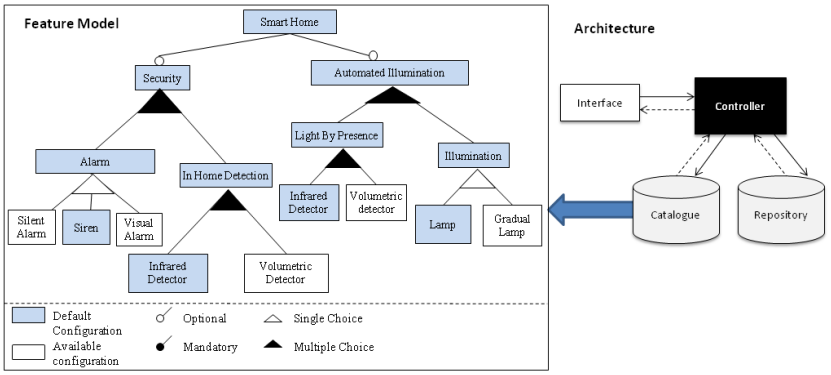


Fig. 1. Elicitation tool architecture

Context scope, System specification, Advanced system and Validation. In the following subsections, we describe the steps that conform each phase in detail.

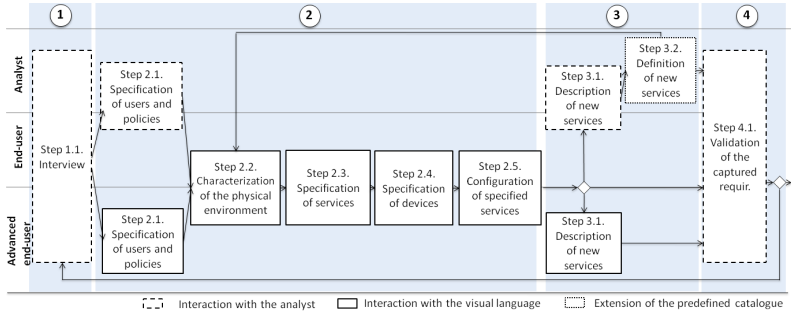


Fig. 2. Natural Requirements Elicitation process

3.1 Context scope

The analyst determines the domain of the system to be developed and the end-user profile. Then, analysts prepare the end-user tool in order to allow end-users to interact with a natural interface adapted to their profile and the system domain. On the one hand, if analysts consider that end-users are familiar with computer applications the tool is prepared to be used through open-option interfaces. Otherwise, the tool is prepared to be used through closed-option interfaces. On the other hand, if, for instance, the system domain is smart homes, the tool is adapted to show only concepts related to smart homes (i.e. locations, services, devices). The analyst captures the information using traditional interviews.

3.2 System specification

End-users must describe the main characteristics of pervasive system by using the presented tool. Five steps are proposed to do this.

Step 2.1: Specification of users and policies. End-users must describe the users who are able to use the system, their desired policies and the links between a user and a policy. For example, two users can be "Paul" and "Mary" while two policies can be "Parents" and "Kids". In further steps, end-users must indicate the services that users of each policy can activate. For instance, users of the "Kids" policy must not be able to activate the security service.

This step is performed in a different way depending on the end-user profile. On the one hand, non-advanced end-users must describe users and policies by working in collaboration with analysts. On the other hand, in order to support advanced end-users to define this information by themselves we have defined a proper interface which is composed of two areas: User and Policy. In each area end-users can manage the users or policies.

Step 2.2: Characterization of the physical environment. Characterization implies to define the different locations in which services can be available and which the system must be aware of. To support this in our tool, we allow end-user to define the different locations that must be known by the system in an image (a map or even a picture) that represents their physical environment. This representation is based on the knowledge and environment they are familiar with. An example of environment is shown in the snapshot of Fig. 3 where the different locations of a home (one corridor, one bathroom, one bedroom, one kitchen and one living room) have been identified over a map.

Step 2.3: Specification of services. End-users have to specify the needed services in their physical environment and where they are located (e.g. Automated Illumination service in the living room). To allow this, we keep the characterization of the physical environment defined in the previous step and we offer the user a predefined catalogue of available services. The tool provides end-users with an interface like Fig. 3 shows: a left frame that offers the list of available services and a right frame that offers the representation of the physical environment. In order to define a service in a location, end-users just need to select it from the left frame and depict it into the proper location. Then, the representative image for the service is displayed.

Step 2.4: Specification of devices. End-users specify the devices that are available in each location. To do this, the tool provides the user with an interface as Fig. 3 shows, which is divided into two frames: the left frame displays the list of devices that are available in the catalogue whereas the right frame represents of the physical environment.

Step 2.5: Configuration of the specified services. End-users are able to define the configuration of each specified service. To support this in our tool, the tool's interface offers a default configuration for each service in a series of left-to-right couplings of jigsaw pieces. Hence to customize a service, end-users have to remove a piece that represents their change and join another compatible piece. For instance, if end-users need to customize the service configuration to use a gradual lamp rather than a lamp they just need to remove the piece representing the Lamp and join the Gradual Lamp piece (see the Service Configuration of Fig.3).

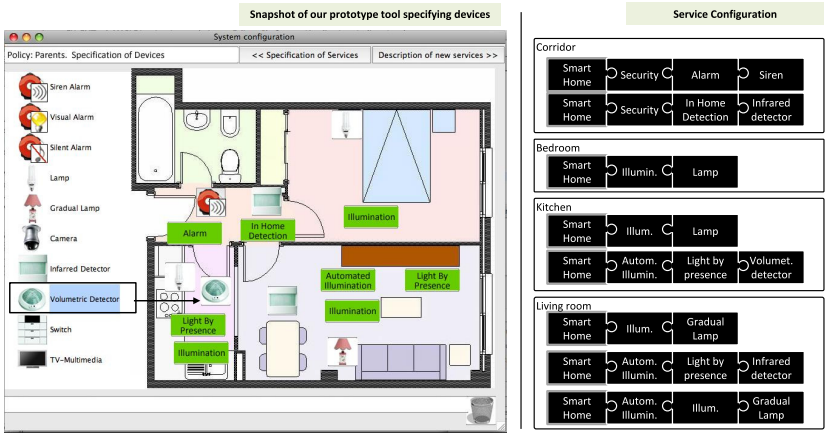


Fig. 3. Requirements for a smart home

3.3 Advanced system configuration

In this phase the information related to the description of new services is captured. Advanced end-users can perform the proposed step by themselves (using an open-option interface) while non-advanced end-users need to interact with analysts.

To describe a new service the needed information is: name of the service, where the service is located, what devices/services are needed to sense the context information (condition) and what devices/services are needed to activate this service (action). For example, end-users may need a Full Automated Illumination service that switches on a lamp and a gradual lamp when presence in the living room is detected. To support this in our tool, the open-option interface offers the same representation of the physical environment introduced above. This visual representation is again complemented with a different left frame that provides advanced end-users with mechanisms to define the information required. In order to introduce information such as location, end-users just need to select the corresponding option in the left frame and then select the location in the representation of the environment.

3.4 Validation

The analyst validates together with end-users the system described by end-users using a traditional interview. In case some ambiguities or mistakes are detected, end-users can perform again the different phases of the process in an iterative way. Otherwise, the obtained description constitutes a valuable product in order to allow analysts to a formal requirement specification that can be used as guide for the rest of the development process. An example of the end-user descriptions obtained by using the proposed tool is shown in Fig. 3. In particular, this figure partially shows the requirements of the smart home system used as a case study.

4 Conclusions and future work

End-users need to be provided with techniques and tools that allow them to specify their needs in a natural way. In this paper, we have presented a natural requirements elicitation technique that helps end-users describe the system they need. In particular, we have focused on the description of pervasive systems by end-users.

To do this, we have presented the prototype of a tool that supports end-users in the creation of pervasive system descriptions. We have also defined a requirements elicitation process that precisely determines the steps that end-users must follow to create a system description. Once the process is finished, analysts have a description which is the base to create a formal requirements specification. As future work, we are working on tools that help analysts to automatically obtain formal requirements specification from these descriptions.

References

1. Brad A. Myers, John F. Pane, and Andy Ko. Natural programming languages and environments. *Commun. ACM*, 47(9):47–52, September 2004.
2. Christopher Scaffidi, Mary Shaw, and Brad Myers. The “55m end-user programmers” estimate revisited, 2005.
3. Andrew J. Ko and Brad A. Myers. Designing the whyline: a debugging interface for asking questions about program behavior. In *CHI ’04*, pages 151–158, 2004.
4. Henry Lieberman. Programming by example (introduction). *Commun. ACM*, 43(3):72–74, 2000.
5. Jan Humble et al. Playing with the bits - user-configuration of ubiquitous domestic environments. In *Proceedings of Fifth Annual Conference on Ubiquitous Computing, UbiComp 2003*, USA.
6. Francisca Pérez, Carlos Cetina, Pedro Valderas, and Joan Fons. Towards end-user development of smart homes by means of variability engineering. In *VaMoS*, volume 29, pages 103–110, 2009.
7. P. Trinidad, D. Benavides, A. Ruiz-Cortés, S. Segura, and A.Jimenez. Fama framework. In *12th Software Product Lines Conference (SPLC)*, 2008.

Parte V

Sesión 4. MDE y Transformaciones

On the Refinement of Model-to-Text Transformations

Salvador Trujillo, Ander Zubizarreta, Josune de Sosa, Xabier Mendialdua

IKERLAN Research Centre, Spain
{strujillo, ander.zubizarreta, jdesosa, xmendialdua}@ikerlan.es

Abstract. *Model Driven Development* (MDD) is a paradigm to automate the generation of code. A key artifact in this paradigm is a model transformation which defines the mappings from a model to another model or even a code artifact. Although MDD was initially aimed at the generation of an individual program, shortly after appeared the need for families of programs. Hence, the combination of Model Driven and Software Product Lines (SPL) appears as a promising paradigm. Most of the previous work was focused on the necessity to support variability on models, but little work has been done so far on supporting the variability of remaining MDD artifacts such as model transformations or metamodels. This work first motivates the need for variability of model transformations. We address then the application of step-wise refinement to model-to-text transformations expressed in MOFScript. We illustrate this with a case study.

Introduction

Modeling is essential to cope with the increasing complexity of current software systems. Models assist developers during the entire development life cycle to precisely capture and represent relevant aspects of a system from a given perspective and at an appropriate level of abstraction.

MDD is a paradigm to automate the generation of boiler-plate code. Raising the abstraction level enables to focus on the domain details and separate the implementation details. This brings a number of specific benefits such as productivity, reduced cost, portability, drops in time-to-market, and improved quality. Overall, the main economic driver is the productivity gain achieved, which is reported by some studies [14, 19].

A key artifact in MDD is a model transformation that defines the mappings between a model and another model or between a model and a code artifact. Although MDD was initially aimed at the generation of an individual program, shortly after appeared the need for families of programs.

Researchers and practitioners have realized the necessity for modeling variability of software systems, where software product line engineering poses major challenges [25]. A software product line is a set of software intensive systems that are tailored to a specific domain or market segment and that share a common set of features [5, 21].

For example, in industrial software systems the presence of different types of subsystems (e.g., exclusive subsystems from different providers) implies that each is controlled in a similar though different way. This is typically achieved by defining two features that are not necessarily present in all possible systems. A feature is an end-user visible behavior of a software system, and features are used to distinguish different software systems or variants of a software product line [16].

Actas JISBD 2009, pp.123-133, ISBN: 978-84-692-4211-7

This impacts not only on the implementation, but on the modeling level. The modeling used in software product lines can be twofold. First, there are approaches for describing the variability of a software product line, e.g, there are feature models that specify which feature combinations produce valid variants [16]. Second, all variants in the product line may have models that describe their structure, behavior, etc.

However, when dealing with variability in an MDD scenario, there are further artifacts apart from models that may need to cope with such variability. Model transformations are a case in point. In some scenarios, they may have too to cope with the variability imposed by the software product line. Hence, this paper takes a step back to study such impact into a broader perspective by analyzing the scenarios for Model Driven Product Lines.

We shift our attention from the variability of models to a more general situation where the variability may embrace models, metamodels and model transformations. Hence, a realization of one feature may consist of variations of such artifacts. This work specifically analyzes the variability of model transformations.

The contribution of this paper is to apply step-wise refinement to model transformations. MOFScript-based model-to-text transformations are refined by using XAK [1]. We illustrate our ideas with a simplistic case study, which is inspired on the industrial cases we work with.

We begin by reviewing the background.

Background

Model Driven Development is a paradigm where models are used to develop software. This process is driven by model specifications and by transformations among models or models and code. It is the ability to transform among different model representations that differentiates the use of models for sketching out a design from a more extensive model-driven software engineering process where models yield implementation artifacts. This paradigm eases cumbersome and repetitive tasks, and achieves productivity gains.

The main artifacts in MDD are models, metamodels and model transformations.

Models. Model is a term widely used in several fields with slightly different meanings. A model represents a part of the reality called the object system and is expressed in a modeling language. A model provides knowledge for a certain purpose that can be interpreted in terms of the object system [17]. Typical examples of models involve statecharts that represents the behavior of some functionality or class diagrams that show the structure of some code.

Figure 1 shows a simple statechart model showing the behavior of a simple temperature sensor. The statechart has states `sClosed` and `sOpened` in addition to initial and final states. Transitions between states and the conditions for these transitions are described in the model. The actions to be executed should be defined too, but are omitted in the example.

Metamodels. A model is frequently considered an instance conforming to a metamodel. A metamodel is a model of a modeling language where the language is specified

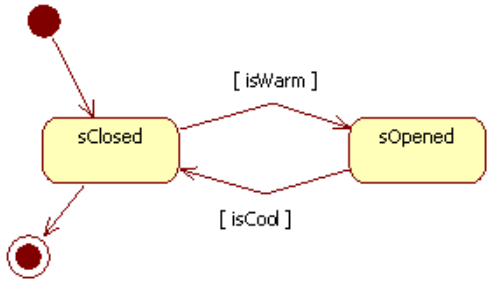


Fig. 1. A simplified statechart for a temperature sensor

[17]. In other words, the metamodel describes the elements of the domain and their relationships.

A metamodel for a statechart might define its elements such as states, transitions, their relationships, etc. Note that the model in Figure 1 might conform to such metamodel. The mappings between metamodels are typically defined by model transformations.

```
1  texttransformation baseTransf (in sc:"http://www.w3.org/2005/07/scxml")
2  {
3    var outputFile: String
4
5    sc.ScxmlType::main ()
6    {
7      file(outputFile)
8      writeSwitchStart()
9      self.state->forEach (s: sc.StateType)
10     {
11       s.generateSwitchCases()
12     }
13     writeDefault()
14     writeSwitchEnd()
15   }
16
17   module::writeSwitchStart ()
18   {
19     /* Empty rule: to be refined later on */
20   }
21
22   /* Content omitted */
23 }
```

Fig. 2. Example of a model-to-text transformation using MOFScript

Model Transformations. Model transformations play a pivotal role in MDD because they turn the use of models for sketching into a more extensive model-driven usage where implementations can be directly obtained [23]. A model transformation

is the process of converting one model to another model of the same system [20]. In general, a model transformation is the process of automatic generation from a source model to a target model, according to a transformation definition, which is expressed in a model transformation language [17].

Depending on the target, model transformations can be model-to-model or model-to-text transformations. The former takes as input one or more models conforming to given metamodels and produces one or more models conforming to the same or another metamodels. The latter produces text as its output, that can be implementation code, documentation, or any other textual form.

Model-to-model transformations usually make use of rules that are defined as mappings between input and output metamodels. Model-to-text transformations combine rules with text templates that define the form of the output text. There is a variety of open-source and commercial tools and languages for model transformations [1].

Our case study focuses on model-to-text transformations defined using MOFScript transformation language. MOFScript language is a metamodel-independent language that allows to use any kind of metamodel and its instances for text generation. MOFScript tool is based on EMF and it is available as an Eclipse plugin.

The definition of a MOFScript transformation consists of rules. Figure 2 shows a base transformation definition for transforming a statechart model into textual code. Implementation-wise, a transformation module called `baseTransf` is defined. Such module is composed by several rules. They can have a context type scoping to which metamodel elements can be applied such rules[2].

On the Need for Variability

A Model Driven Development scenario typically involves models, metamodels and model transformations. Next, we analyze the need for variability of model transformations.

Scenario. There are different scenarios when combining MDD and SPL. Consider the differences on the modeling language used: it is not the same to use UML or a Domain Specific Language (DSL).

There are scenarios where model variability may be enough and the variability of model transformations may not be needed. This may happen in situations where the used metamodel is standard and so it is not subject to variability. For instance, when using a UML class diagram, it seems unlikely to make its metamodel variable since it is somehow standardized. This may apply generally to the metamodels within the UML. A similar situation occurs when the model transformations come from a common library

[1] ATL [4], RubyTL [7], ATC [24] and other tools are for model transformations. Though these tools may be also used (and so we do) for model-to-text transformations, we focus on MOFScript since it is specialized on model-to-text.

[2] The execution of the transformation in Figure 2a starts with the `sc.ScxmlType::main()` rule. This rule is applied to the context type scoped by the type `ScxmlType` where `sc` is the input model. That means that it applies to every `ScxmlType` in `sc`. The output text file is defined with `file()`.

of model transformations that are shared. For instance, consider the dozens of model transformations expressed in ATL that are available online[3].

Therefore, in scenarios with standard metamodels and/or shared transformations, the use of model variability seems enough and thus variability of model transformations may not be needed. However, there exist other scenarios where the variability of model transformations may be needed.

Consider the case in which different models and model transformations may need to be customized for different targets. Model transformations share a significant common part while differing in some variable parts. For instance, different implementation code can be generated from the same source model. The target code is expected to be executed in different platforms with different programming languages. This situation could be well handled by defining features of the software product line. There is a large proportion of shared code and some particularities are bound to each programming language. In such situation, the application of variability to model transformations may enable to handle those differences in a unique model transformation. Next, we illustrate this with our case study.

Case Study. Although this motivating scenario is realistically more likely to occur within a larger system, we illustrate our ideas with a family of control software systems developed following MDD. Particularly, we focus on the control of a simple temperature sensor that opens/closes some device (e.g. water pump). Our motivating scenario demands to cope with the variability of models, metamodels and model transformations. The files used in this example are available online[4].

Consider a model of a simple state machine and a model-to-text transformation definition to get implementation code from it. The aim is to generate a switch statement implementing the statechart model. There are however different target platforms for the generated code. Such different programming languages are Ada and Java. Switch statements differ from Ada to Java. Actually, Ada does not provide a proper switch statement, but a case statement with equivalent structure and behavior to Java's switch.

It would be possible to define a separated and independent model transformation definition for each target language. In that way, however, it would be necessary to define a new transformation definition for each new target language we would like to generate code for. Doing so, no reuse will be achieved.

The structure of the switch statement is similar in most of the programming languages. Its difference mostly lies in the syntax. Hence, it would be possible to define a base transformation common for all target platforms and to refine it with partial or refined transformations incorporating target specific elements. This is what we called in this work as the refinement of model-to-text transformations.

Refinement-based Variability

The refinement of model-to-text transformations is introduced in order to enable their customization for different variability needs. In our example, there is a statechart as

[3] http://www.eclipse.org/m2m/atl/atlTransformations/

[4] http://www.ikerlan.es/softwareproductline/jisbd2009examples.zip

a source model, Ada and Java as target programming languages and a model-to-text transformation with variability incorporated to connect the source with different targets.

As a case in point, we considered the generation of switch statements that implement the code of a statechart model. Although switch statements are commonly found in most of current programming languages, they are not exactly defined in the same way. In our example, we define a base transformation with all the common elements of the switch, that will be refined for each output language. There will be a transformation refinement associated to each target language, each adding its specific elements.

Base of a Transformation

```
1  <MOFScriptModel:MOFScriptSpecification ... xak:artifact="base.m2t.model.mofscript"
       xak:feature="baseTrans">
2    <transformation line="6" name="baseTransf" xak:module="transModule">
3      <variables line="7" column="2" name="outputFile" type="String" xak:module="
           oFileModule"/>
4      <parameters line="6" column="39" name="sc" type="http://www.w3.org/2005/07/
           scxml"/>
5      <!--Content omitted-->
6      <transformationrules line="64" name="writeSwitchStart" xak:module="
           switchStartModule">
7        <context line="64" name="self" type="module"/>
8      </transformationrules>
9      <!--Content omitted-->
10   </transformation>
11 </MOFScriptModel:MOFScriptSpecification>
```

Fig. 3. Base Transformation (XMI representation)

A base transformation definition has been defined with MOFScript (see Figure 2). This definition takes a statechart model as input, and has the rules to generate a switch statement as output. Since the switch statement differs among different languages, this base transformation only has the common elements.

In our example, the base transformation defines some common rules to generate a switch statement based on the statechart model. Some rules for writing the output are defined (e.g. `writeSwitchStart`) but they are empty due to the fact that the output text varies depending on the syntax of the output programming language. In the base transformation a variable named `outputFile` is defined too, to specify the name of the output file, but this variable is not yet given a particular value. The assignment will be done with a refinement, which will assign a name related to its output language later on.

MOFScript allows to represent a transformation definition as a model, conforming to the MOFScript metamodel. Figure 3 shows the model representation of the transformation using XML Metadata Interchange (XMI). (Note that this figure is equivalent to the textual representation of the transformation in Figure 2.) The transformation module is defined with the `<transformation>` element. Nested to this element, variables, parameters and rules are defined using the `<variables>`, `<parameters>` and `<transformationrules>` elements, respectively.

Actually, the base transformation can not be executed standalone since it is not complete, so it may not generate the expected implementation. The base transformation needs to be refined for each output language.

Refinement of a Transformation

```
1  <xak:refines xak:artifact="base.m2t.model.mofscript" xak:feature="adaDelta" ...>
2    <xak:extends xak:module="oFileModule">
3      <xak:super xak:module="oFileModule"/>
4      <value xsi:type="MOFScriptModel:Literal" value="example.adb"/>
5    </xak:extends>
6    <xak:extends xak:module="switchStartModule">
7      <xak:super xak:module="switchStartModule"/>
8      <statements xsi:type="MOFScriptModel:PrintStatement">
9        <printBody xsi:type="MOFScriptModel:Literal" value="case␣state␣is&#xA;&#x9
             ;&#x9;"/>
10     </statements>
11     <blocks statements="//@transformation.0/@transformationrules.3/@statements.0"
           protected="true"/>
12   </xak:extends>
13   <!—Content omitted—>
14 </xak:refines>
```

Fig. 4. XAK-based Refinement of a Transformation

A refinement can be deemed of as a function that takes an artifact as an input, and returns another similar artifact which has been leveraged to support a given feature [3]. XAK is a language for defining refinements in XML documents and provides a composer tool for them [1]. We used it for representing and composing our MOFScript-based transformations. MOFScript offers the option to represent a transformation definition as a model. Such model can be represented using XMI. Therefore, XAK can be used for refining such transformations.

Any traditional XML document can be a base document, but some additional annotations are needed (see Figure 3). The attributes `@xak:artifact` and `@xak:feature` are added to the document root element. The first one specifies the name of the document that is being incrementally defined, while the second one specifies the name of the feature being supported (“base” for base documents). To specify which elements are modularizables, `@xak:module` is used. That is, it indicates those elements that play the role of modules, and henceforth can be refined. In general, a XAK module has a unique name.

Refinement documents refine base documents and they have `<xak:refines>` element as root (see Figure 4). Its content describes a set of module refinements (i.e. elements annotated with the `xak:module` attribute) extending a given base document (i.e. the `xak:artifact` attribute). The `xak:super` node is a marker that indicates the place where the original module body is to be substituted. In general, a XAK refinement document can contain any number of `xak:module` refinements.

Those elements of the transformation that can be refined must be defined before refining such transformation. In our example, the refinement units are the transforma-

tion definition itself, variables and rules. That is, we can add new rules or variables to the transformation definition, we can add or set values to variables already defined, or we can extend existing rules with new statements. We add `xak:module` attribute to `<transformation>`, `<transformationrules>` and `<variables>` elements to specify that they are modularizables (see Figure 3). Note that not all elements are modularizable, but only those we designate beforehand by `xak:module`.

Figure 4 shows a refinement for the base transformation. This refinement refines the base transformation definition to generate the code in Ada programming language. Some of the rules in the base transformation are refined adding Ada specific elements. The variable `outputFile` is given a value. The rules that are refined are those which are used to produce specific code to the output file[5]. Since the syntax differs between languages, those rules must be refined for each language.

Composition of Base and Refinements

```
1  <MOFScriptModel... >
2    <transformation ...>
3      <variables ...>
4        <value value="example.adb" .../
             >
5      </variables>
6      <!--Content omitted-->
7      <transformationrules ...>
8        <context .../>
9        <statements ...>
10         <printBody value="case␣state␣
               is&#xA;&#x9;&#x9;" .../>
11       </statements>
12       <blocks .../>
13     </transformationrules>
14     <!--Content omitted-->
15   </transformation>
16 </MOFScriptModel...>
```

(a)

```
1  /**
2   * Transformation Generated by
        MOFScript2Text transformation
3   */
4  texttransformation baseTransf (in sc
       :"http://www.w3.org/2005/07/
       scxml"){
5      var outputFile: String = "example
           .adb";
6
7  /* Content omitted */
8
9  module::writeSwitchStart () {
10 "case state is\n    "
11 }
12
13 /* Content omitted */
14 }
```

(b)

Fig. 5. Composed Transformation: (a) XMI representation; (b) MOFScript representation

XAK composer tool is used to compose the refinement to the base transformation, executing the following command:

> xak -xak2 -c baseTrans.mofscript adaDelta.xak -o composedAdaTrans.mofscript

The result of the composition is a composed transformation model (Figure 5a), which can be executed directly giving as input the statechart model, or can be converted

[5] Due to the lack of space, in the figure only the refinement of the rule `writeSwitchStart` is shown. The complete example is available for download at http://www.ikerlan.es/softwareproductline/jisbd2009examples.zip

```
case state is
  when sClosed =>
    if isWarm then
        state := sOpened;
    end if;
  when sOpened =>
    if isCool then
        state := sClosed;
    end if;
  when others =>
end case;
```

(a)

```
switch (state) {
  case sClosed:
    if (isWarm) {
      state = sOpened;
    }
    break;
  case sOpened:
    if (isCool) {
      state = sClosed;
    }
    break;
  default:
}
```

(b)

Fig. 6. Generated code: (a) Ada; (b) Java

to a MOFScript transformation definition file (Figure 5b) using that option in MOFScript tool and then executed. Note that some rules that were empty in the base transformation definition, have now some body specifying how the output has to be in Ada programming language. So, the customization of model transformations is achieved. The output of the transformation is the `example.adb` file which has the switch statement according to our model, written in Ada syntax (Figure 6a).

Another possible refinement may be to refine the base transformation definition to generate Java code. In this case, the rules should be refined to generate Java code. Although the structure of the generated code is almost the same, the syntax differs from Ada to Java. Additionally, `break;` sentences are necessary in Java, so a new rule needs to be added to the base transformation to write such sentences. The new rule will be called from `generateSwitchCases` rule, that need to be refined to add the new statement. Figure 6b shows the code obtained when running the transformation refined for Java.

The use of refinements in the context of model-to-text transformations enables the reuse and customization that Software Product Lines promote.

Related Work

Merging MDD and product lines is not new, we know of examples that explicitly use features in MDD [9, 8, 10–12, 22]. One is BoldStroke: a product-line for supporting a family of mission computing avionics for military aircraft [12]. Czarnecki introduces super-imposed variants and model templates to map features to models [8]. Weber et al. introduce the Variation Point Model that models variation points at the design level [26].

There is a line of work on feature-based composition of models. Feature-oriented model-driven development is an approach that ties feature composition to model driven development [25]. Recent work by Apel describes superimposition as a model composition technique to support variability of product lines [2]. FeatureMapper is a tool that

supports mapping features from feature models to solution artifacts [13]. These works do not yet consider the composition of model transformations or metamodels.

Sanchez-Cuadrado presents an approach for the reuse of model transformations in RubyTL by using an idea reminiscent of libraries in programming [6]. This first step towards model transformation reuse does not yet incorporate the notion of product family. The superimposed modules of ATL language can be composed into different transformation definitions. This is not related to features, neither to the notion of composition demanded in a product family scenario [15]. Oldevik proposes an aspect-based extension of the MOFScript model-to-text transformation language, which is called a Higher Order Transformations (HOT) [18].

Conclusions

This paper presented an approach for the variability of MOFScript-based model-to-text transformations in a software product line scenario. The main contribution is the application of step-wise-refinement in the context of a model transformation. Doing so, the application of variability shifts from scenarios focused only on model variability towards broader scenarios handling the variability of MDD-related artifacts.

We claim the need to shift our research attention from the variability of models to a broader perspective embracing the variability of models, metamodels and model transformations.

This is indeed the direction of our current research efforts where we are addressing the variability of models, metamodels, and model transformations and their relationships concomitantly, since we believe they are often closely inter-related.

Acknowledgments. This work was co-supported by the Spanish Ministry of Science & Innovation under contract TIN2008-06507-C02-02. We thank to Maider Azanza for her comments on earlier drafts.

References

1. F. I. Anfurrutia, O. Díaz, and S. Trujillo. On Refining XML Artifacts. In *ICWE*, pages 473–478, 2007.
2. S. Apel, F. Janda, S. Trujillo, and C. Kaestner. Model Superimposition in Software Product Lines. In *2nd International Conference on Model Transformations (ICMT 2009), Zurich, Switzerland, June*, 2009.
3. D. Batory, J.Neal Sarvela, and A. Rauschmayer. Scaling Step-Wise Refinement. *IEEE Transactions on Software Engineering (TSE)*, 30(6):355–371, June 2004.
4. J. Bézivin, G. Dupe, F. Jouault, G. Pitette, and J. E. Rougui. First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery. In *2nd OOPSLA Workshop on Generative Techniques in the context of MDA, Anaheim, California, USA, Oct 27*, 2003.
5. P. Clements and L.M. Northrop. *Software Product Lines - Practices and Patterns*. Addison-Wesley, 2001.
6. J. Sanchez Cuadrado and J. Garcia Molina. Approaches for Model Transformation Reuse: Factorization and Composition. In *International Conference of Model Transformations (ICMT)*, 2008.

7. J. Sánchez Cuadrado, J. García Molina, and M. Menárguez Tortosa. RubyTL: A Practical, Extensible Transformation Language. In *2nd European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2006), Bilbao, Spain, Jul 10-13*, pages 158–172, 2006.
8. K. Czarnecki and M. Antkiewicz. Mapping Features to Models: A Template Approach Based on Superimposed Variants. In *4th International Conference on Generative Programming and Component Engineering (GPCE 2005), Tallinn, Estonia, Sep 29 - Oct 1*, 2005.
9. K. Czarnecki and M. Antkiewicz. Model-Driven Software Product-Lines. In *20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2005), San Diego, CA, USA, Oct 16-20*, 2005.
10. S. Deelstra, M. Sinnema, J. van Gurp, and J. Bosch. Model Driven Architecture as Approach to Manage Variability in Software Product Families. In *Workshop on Model Driven Architecture: Foundations and Applications (MDAFA), Enschede, The Netherlands, June 26-27*, 2003.
11. B. Gonzalez-Baixauli, M.A. Laguna, and Y. Crespo. Product Lines, Features, and MDD. In *1st Europeean Workshop on Model Transformation (SPLC-EWMT'05), Rennes, France, Sep 25*, 2005.
12. J. Gray and et al. Model Driven Program Transformation of a Large Avionics Framework. In *3th International Conference on Generative Programming and Component Engineering (GPCE 2004), Vancouver, Canada, Oct 24-28*, 2004.
13. F. Heidenreich, J. Kopcsek, and C. Wende. FeatureMapper: Mapping Features to Models. In *30th International Conference on Software Engineering (ICSE 2008), Companion*, pages 943–944, New York, NY, USA, may 2008. ACM.
14. D. Herst and E. Roman. Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) - Approach: A Productivity Analysis. Technical report, TMC Research Report, 2003.
15. F. Jouault and I. Kurtev. Transforming Models with the ATL. In *International Conference on Model Driven Engineering Languages and Systems (MODELS 2005)*, 2005.
16. K. C. Kang and et al. Feature Oriented Domain Analysis Feasability Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, November 1990.
17. I. Kurtev. *Adaptability of Model Transformations*. PhD thesis, University of Twente, 2005.
18. J. Oldevik and O. Haugen. Higher-order transformations for product lines. *Software Product Line Conference, International*, 0:243–254, 2007.
19. OMG. MDA Success Stories. http://www.omg.org/mda/products_success.htm.
20. OMG. MDA Guide version 1.0.1. OMG document 2003-06-01, 2003.
21. K. Pohl, G. Bockle, and F. van der Linden. *Software Product Line Engineering - Foundations, Principles and Techniques*. Springer, 2006.
22. D. Schmidt, A. Nechypurenko, and E. Wuchner. MDD for Software Product-Lines: Fact or Fiction. In *Workshop at 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005), Montego Bay, Jamaica, Oct 2-7, 2005*, 2005.
23. S. Sendall and W. Kozaczynski. Model Transformation: The Heart and Soul of Model-Driven Software Development. *IEEE Software*, 20(5):42–45, 2003.
24. A. Sánchez-Barbudo, E. V. Sánchez, V. Roldán, A. Estévez, and J.L. Roda. Providing an Open Virtual-Machine-based QVT Implementation. In *Proceedings of the V Workshop on Model-Driven Software Development. MDA and Applications (DSDM'08 - XIII JISBD)*, 2008.
25. S. Trujillo, D. Batory, and O. Díaz. Feature Oriented Model Driven Development: A Case Study for Portlets. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May*, 2007.
26. D.L. Webber and H. Gomaa. Modeling Variability in Software Product Lines with the Variation Point Model. *Science of Computer Programming*, 53, 2004.

Towards Automatic Code Generation for EAI Solutions using DSL Tools *

Hassan A. Sleiman[1], Abdul W. Sultán[2], Rafael Z. Frantz[3], Rafael Corchuelo[1]

[1] Universidad de Sevilla, ETSI Informática
Avda. Reina Mercedes, s/n. Sevilla 41012
hassansleiman@us.es, corchu@us.es
[2] Sytia Informática S.L.
Avda San Sebastián, nº10, Local 1, Entreplanta. Huelva 21004
awsultan@sytiainformatica.es
[3] Universidade Regional do Noroeste do Estado do Rio Grande do Sul (Unijuí)
Departamento de Tecnologia, São Francisco, 501. Ijuí 98700-000 RS Brasil
rzfrantz@unijui.edu.br

Abstract. Current companies count on heterogeneous information technology applications to perform their activities. More often than not, they need to be integrated so that the data they manage is kept in sync or to implement new functionality. According to a recent report by IBM, companies spend from $5 to $20 on integration per dollar spent on developing new functionality. This ratio argues for engineering solutions. The Model-Driven Architecture initiative promotes the development of software systems at different levels of abstraction, and Domain Specific Languages (DSLs) play a prominent role to reduce development costs. By means of an appropriate DSL, software engineers can design a software system that can later be deployed to a variety of specific platforms using automatic transformations. Our proposal to reduce integration costs is a DSL called Guaraná and a software tool to design and automatically deploy integration solutions. Compared to the Enterprise Application Integration UML-profiles, DSLs are more suitable to address problems in a particular domain and are a better approach towards MDA.

Key words: Enterprise Application Integration, Domain Specific Language.

1 Introduction

We define an integration solution as a piece of software that co-ordinates a number of applications exogenously. An integration solution may focus on: Enterprise Application Integration (EAI); Business to Business Integration (B2BI); Enterprise Information Integration (EII); Extract, Transform and Load (ETL) and Mashup. Our research focuses on EAI.

EAI always happens inside the same company, in its own software ecosystem, and the goal is to keep applications synchronized or to create new functionalities on the top

* Partially funded by the Spanish National R&D&I Plan under grant TIN2007-64119, and the Andalusian Local Government under grants P07-TIC-02602, and P08-TIC-4100. The work by R.Z. Frantz was also funded by the Evangelischer Entwicklungsdienst e.V. (EED).

of them. Furthermore, when considering Business to Business Integration, the solution necessarily involves applications that belong to different companies, so different software ecosystems must be considered. This type of solution has the same aim of EAI, but other aspects when building the integration solution must be considered, e.g. authentication, external network communication failures, service availability, confidentiality, non repudiation or accountability must be taken into account. In addition, standards like Open Buying on the Internet (OBI), Electronic Document Interchange (EDI) and Commerce XML (cXML) are usually adopted for the communication amongst companies. Also, it is very important to point here that those pieces of the integration solution deployed inside a software ecosystem (or in case we deploy it as a whole, so the whole solution) will have unlimited access to the application's layers that are part of the same ecosystem; to communicate with external applications that take part of the integration solution, but belong to other companies' software ecosystems, it must be done only by means of the exposed public interfaces of those external applications.

According to a recent report, for each dollar spent on developing an application, companies usually spend from 5 to 20 dollars to integrate it [11]. This claims for solutions that can contribute to reduce this high cost of integration. Guaraná is a domain specific language to design integration solutions that aims to simplify the designing process of solutions and produces platform independent models which can programmatically be transformed into code of a specific technology [6]. In this paper, a realistic integration problem was taken as motivation and validation to generate an application integration solution using Guaraná's language and Windows Workflow Foundation (WF) [4] as target deployment technology. In order to meet this goal we have created a software tool prototype that implements domain specific concepts defined in Guaraná and thus allow the visual design of integration solutions. The platform-independent model designed with this software can be transformed into several integration technologies; however, in this paper, we focus on Windows Workflow Foundation.

Actually, [7] describes patterns for EAI but no integration solution language is defined, while Camel and Biztalk provide languages to describe integration solutions. In the case of Camel, this language is textual whereas Biztalk, although it is a graphical language, its not based on MDA. We understand that a DSL is a well focused language developed to address problems in a particular domain providing a set of dedicated abstractions, elements and notations with formalisation to assist the designer in expressing its solution in the idiom and at the level of abstraction of its DSL. We are aware of the existence of an EAI UML-profile [9] proposed by OMG as an extension for their UML to support some concepts of EAI. While DSL is usually a small and well focused language, UML-profiles intend to wide the scope of UML to cover the modelling of those specific aspects not covered by the native UML elements [1]. This extension has some limitations, such as the impossibility to introduce new modeling concepts that cannot be expressed by extending existing UML elements. These characteristics contributes to make UML-profiles more complex and difficult to use in some domains. This profile also seems to be discontinued.

The transformation of an integration solution designed using Guaraná into Windows Workflow Foundation (WF) is just one among many options, since there are many other possible target integration technologies like those discussed by authors in [5]. These au-

thors also have designed a framework that can be used to compare different integration technologies and they provide a comparison of five technologies in [3]. In [6] authors use 15 properties of this comparison framework to evaluate the Guaraná's language against some most common integration technologies' language.

This paper is organised as follows. Section 2, introduces the domain specific language tools concept and briefly highlights important concepts of Guaraná; Section 3, presents an integration problem from the University of Ijui (UNIJUÍ) used to validate our software tool; Section 4, presents the software tool prototype; Section 5, introduces how the transformation process to Windows Workflow Foundation is carried out; and, finally, in Section 6, we draw our conclusions.

2 Guaraná, a DSL for EAI

Domain Specific Languages (DSLs) are modelling languages designed to be used in special types of problems. They are restricted to a domain and are characterised by a high-level of abstraction that allows to express the concepts in the language of the problem domain. Some examples of traditionally used DSLs include: SQL (for database access), HTML (to describe the structure and content of a document) and BNF (to describe context-free grammars).

The DSLs listed above are textual languages that are specified by describing the language syntax (eg. using BNF), which requires a parser for the language. There are also graphical DSLs based on a set of graphic symbols. Such DSLs have a direct correspondence with the conceptual model. Besides, the description of these languages can be done graphically. Considering the benefits of graphical DSLs, we have relied on this kind of DSL to perform this work.

The most important aspects of a graphical DSL are: domain model, notation and code generation. The domain model is a model of concepts described by the language. The basic constructors in graphical DSLs are domain classes and domain relationships. Domain classes represent the concepts of the problem domain whereas domain relationships represent relationships between these concepts. In addition, each domain class and domain relationship contains a set of properties. An important aspect of the domain model is the definition of constraints that the model must satisfy. These constraints are used to verify that the created diagrams are valid. The notation is a set of graphical symbols used to represent the domain model. The basic elements are shapes and connectors that are the graphical representation of the domain classes and the domain relationships. There is a direct correspondence between domain classes and domain relations and its graphical representation. This graphical notation is used to edit the model.

2.1 Guaraná

Guaraná provides a set of domain specific constructors to design integration solutions, which are described in the language's metamodel [6], where a part of them are inspired from the patterns at [7]. This language provides a very expressive and needful graphical notation for these constructors, which allow us to visually design an integration solution. Below we introduce the main constructors that can be seen in Fig. 1.

Building Block: This is one of the most important concepts in our language, since it represents a general constructor block where most of an integration solution processing takes place. Although some building blocks receive no entry messages, a typical building block receives an inbound message, executes one or more atomic tasks, and then makes the resulting message available for the next element(s)in the flow. Apart from being composed of tasks, building blocks have ports through which they receive and send messages. Basically, there are two kinds of building blocks: Wrappers and Processes. Wrappers are used to connect an application to an integration solution. Therefore, necessarily, one of its ports are connected with a specific application. Its internal tasks prepares messages to be sent to the application and/or to other processes of the integration solution. On the other hand, processes represent internal blocks where a well-defined set of tasks perform a clear integration service of the solution. A process is connected to other processes or wrappers by means of ports and integration links.

Task: Is the element responsible for a building block's internal processing and allows to turn a process or wrapper into a more complex processing unit. A task reads a message from a slot, processes it and writes the result to the next slot, making it available for the next element inside the block. This message processing usually consists of executing an integration pattern, e.g. enriching, translating, filtering or routing.

Slot: They are used inside building blocks to allow exchanging messages between ports and tasks, and also between tasks,ie., they are essentially buffers.

Port: These elements abstract away from the method used to communicate building blocks and/or applications. Guaraná provides four types of ports, divided into two categories: one-way ports and two-way ports. The former are used for internal or external communication whereas the latter are used for external communication with an application. One-way ports are divided into entry port and exit port; two-way ports are divided into solicit-responde port (solicitor port) and request-response port (responder port). An entry port reads information from an application by accessing one or more of its layers. The possible layers we consider here are: data layer, business layer, controller layer and graphical user interface layer. An exit port does the opposite of an entry port, that is, it writes information to an application's layer. A solicitor port enables the integration solution to solicit information from an application. A responder port provides a request-response interface that an application can use to send requesting messages and receiving responses from the integration solution.

Integration Link: Used internally in an integration solution as a mean to transport messages from one building block to another. Because ports represent entry/exit points at a building block, integration links are those elements that actually connect them.

3 Validation example

We have validated our proposal by designing and generating code for an integration problem found at UNIJUÍ. The goal was to automate the invoicing of the calls not related to te employee's work activities. At UNIJUÍ, five applications involved in a handcrafted process to invoice their employees of the private phone calls they make using the University's phones. Applications run on a different platform and were designed without integration concerns in mind. There is a Call Center System (CCS) that records

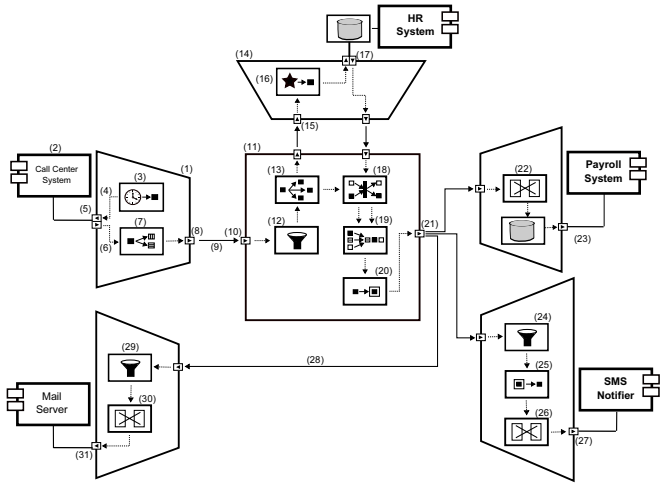


Fig. 1. UNIJUÍ's integration solution.

every call every employee makes from one of the telephones this university provides to them. Every month, an analysis is performed to find out the price of the private phone calls; such calls are debited to the employees by using a Payroll System (PS). There is also a Human Resources System (HRS) that provides personal information about the employees, including their names, phone numbers, social security numbers, and so forth. There are two additional systems, a Mail Server (MS) and an SMS Notifier. Figure 1 depicts these applications and our integration solution.

The integration flow for the solution presented in this example begins with a timer task (3) inside the wrapper (1) of application CCS (2). This task creates an activation message every five minutes and writes it to the slot (4) that connects it to the solicitor port (5), which then is accessing. This message activates the solicitor port that extracts all those calls which were made in the last five minutes. The only way to communicate with this application is by means of its user interface since no clean interface is provided, so the solicitor port will have to use a scrapper [2] to perform its work. The port will return a big message that contains, probably, many phone calls and then writes it to the next slot (6) in the flow. At this point a splitter task (7) is used to break the incoming message into new messages that contains just one phone call each. The exit port (8) of this wrapper reads each message from its previous slot and then sends it to the integration link (9), making it available for the entry port (10) of the central process block (11). This process starts with a filtering task (12)which filters out messages that do not have a cost for the university, and allow just toll calls to remain in the flow. Those messages are written to the next slot, and will be read by the next task, a replicator (13). The replicator makes copies of the original message. In this case one copy is sent to the wrapper of the application HRS and the other to the next element in the current process. In this integration solution we need to append missing information about the employee to the message, like: name, department, e-mail and mobile phone, and for this purpose the HRS, that contain this information, is also integrated into our solution. The message

copy received by the wrapper of HRS, through the entry port (15), will be processed by a custom task (16). This task produces an outbound message that represents a database query to be executed against the database using the solicitor port (17) of this wrapper. After that when the correlation set task (18) receives the result from the HRS's wrapper, it gives pass to the result and the original message that was waiting inside one of its entry slots. The next task, a merger (19) reads the two correlated messages and writes them to the single entry slot of the enricher task (20), so it enriches the original message with the result from the HRS. Now the enriched message is sent to the next slot, the one that connects with the exit port (21). This port is also connected to three integration links that allow sending a message copy to PS, SMS and MS.

The first integration flow after the exit port (21) connects the process (11) to the wrapper of the PS application. This wrapper receives the message through its entry port and makes it available for the translator task (22). The translator is responsible for translating the current message format into a new format that the PS can understand, and so the exit port (23) of the wrapper writes the message into the application's database. The second flow connects the same exit port (21) to the SMS' wrapper. This wrapper has a filter task (24) to filter out those messages that, for some reason, does not have the employee's phone number and then those messages that could pass are sent to a slimmer task (25). A slimmer is responsible for removing some information from the message in order to make it smaller before sending it to the SMS. The SMS is an external application that allows sending messages to mobile phones. The last task in this wrapper is a translator (26) that receives the inbound message and translates it into a special format that the SMS can understand. Once the SMS offers a public gateway the exit port (27) will interact with it through remote procedure calls to forward the message.

The last copy of the message goes to the flow (28) that now connects the process (11) with the wrapper of the MS. This wrapper integrates the application allowing the solution to send e-mails with all the details about the employee's call. The first task in this wrapper is a filter (29), here again to filter out messages that for some reason does not have the employee's email address. As in the other wrappers it is important to translate the inbound message into a message format that the MS can understand. This is done using a translator (30) inside the wrapper, just before its exit port (31). The translated message now goes, through a port that uses remote procedure call to communicate with the application's gateway.

4 A DSL tool for EAI

Microsoft DSL Tools (MS/DSL Tools) is a framework that eases the development of DSLs and graphical editors for them. The framework consists of a project wizard to create configured solutions, a graphical designer for defining and editing domain models, designer definitions in XML, a set of code generators that produce code that implements domain model definition, a designer definition and a framework for defining text output generators [8].

Domain models in MS/DSL Tools are defined using class hierarchies and identifying relationships between theses classes. Relationships can be either reference or com-

position. First, we define Guaraná's metamodel and once defined, we proceed to create this metamodel in MS/DSL Tools. IntegrationSolution is the root class of integration metamodel. This class represents the integration solution. An integration solution designed with MS/DSL Tools is an instance of this class.

An Integration Solution is composed of IntegrationSolutionBlocks and Applications. A composition can be represented in MS/DSL Tools by an embedding relationship. There are more composition relationships in the integration metamodel. A block in an integration solution is composed of EntryPort and ExitPort that allow the connection of IntegrationSolutionBlocks which are composed of a set of tasks whose functionalities were described before.

Another kind of relationships is the inheritance. In our integration metamodel, BuildingBlock and Hub are classes that derive from IntegrationSolutionBlock, while Wrapper and Process derive from BuildingBlock. Further, all the kinds of tasks inherit from Task. Unlike composition, reference relationships do not have any limitation on the multiplicities of each of their roles. This allows us to connect any type of class to create graphs with really complex models.

Entry ports and exit ports are always bound with one another through integration link relationship. Applications are connected with their wrappers through application link relationship. Other reference relationships in integration metamodel are slots that interconnect tasks to each other. After creating the above metamodel, shortly explained here, we obtain Guaraná's Designer, cf. Figure 2.

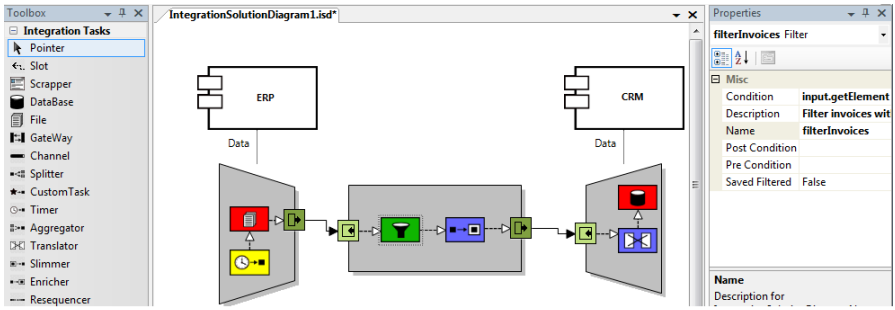


Fig. 2. An integration solution created in Guarana's DSL tools.

Guaraná's Designer consists of a drawing area where integrations solutions can be modelled and a toolbox that allows us to drag and drop elements of Guaraná's DSL in an integration solution. Guaraná's DSL tool has also a properties window where we can edit the properties of ports, tasks and building blocks.

In the toolbox, we can find Process, Application and Wrapper which are elements that can be added directly to the model. Tasks are divided into four groups (Router, Constructor, Interfacing and Transformer) represented by different colors (green, yellow, red and blue respectively). These tasks can only be created within a process or a wrapper. Entry Ports and Exit Ports which also apply to the Processes and Wrappers. We can find three types of connectors: Integration Links to connect with Entry Ports and

Exit Ports, Slots for connecting Tasks and Ports within the Processes and Wrappers, and Application Links to connect applications and wrappers.

5 Automatic code generation

Our proposal is to transform our designed integration solutions into workflows using Microsoft Windows Workflow Foundation (WF), and then make use of the WF's runtime to run the generated workflow. From our point of view, an integration process can be seen as a business process where information flows between participants.

There is no doubt that the appearance of workflows has changed the way processes are done. The partial or complete automating of a business process is called workflow, where documents, information or tasks flow between the participants under some constraints and rules, and where the main goal is to achieve a business goal, such as client satisfaction. We can define workflows as the movement of information through a business process between resources [10]. First, we transform our integration solution into a workflow, then we use WF as a workflow management system in which the generated workflow is executed and monitored.

5.1 Transformating our DSL into WF

Transformation is achieved by the creation of three files: a designing file where the workflow is described (Xoml in this case), a code file where tasks' functionalities are described (C#), and a third file that will host and launch the created workflow (C#). The basic unit in an integration solution is the task, while the basic unit in a workflow is called activity. In this transformation, every task of the integration solution is transformed into one or more workflow activities. The resulting activities represent the functionality of the original task in our integrations solution, but some limitations might not allow us to implement the complete functionality. We start explaining the implemented transformations and then we describe the limitations we found.

System messages: To model system messages, we used XML. A message in our system is an XML file, which we chose for the easiness of its management and the large number of APIs that can be used to manipulate this type of files. It is an instance of the class XmlDocument of the .Net API from Microsoft. Messages inside our system do not have a predefined scheme since they are produced by the applications we are integrating. It is user's responsibility to transform the input messages of an application to another format understandable by another application using a task from the transformers group.

Wrappers: Inside a wrapper, we can find tasks from the interfacing group that are used to communicate with the applications we are integrating, but it can be seen as a process too, where the first activity reads information from an application and then it is processed and written to an exit port.

Processes: An integration process can be seen as a business process. Since WF is oriented to business processes, we were able, although with some limitations, to transform integration processes into workflows. To achieve this, entry and exit ports, tasks and slots that interconnect these tasks were transformed into workflow elements.

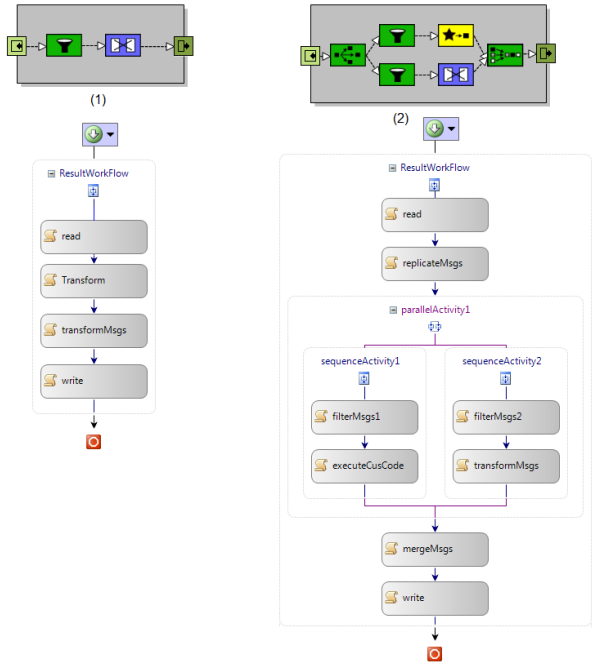


Fig. 3. Integration processes seen as workflows in WF.

Ports: Ports are simulated by activities that read from input directories and write to output directories. An entry port is an activity that reads XML files from a directory to inject them into the process so that they flow between processes's tasks. Exit ports are activities that write messages to an output directory. We used .Net features to monitor directories so that a process can be activated only when one or more messages are detected in the entry port (input directory). When a port reads a message, it is consumed and disappears from the port.

Slots: In an integration solution, a slot is totally different from arrows and links connecting workflow activities. In workflows, an arrow indicates that the origin task is a predecessor and the target task is a successor. For this reason, the responsibility of writing resulting messages in the target tasks were assigned to the task that produces the messages due to this difference in concepts. A predecessor task has a reference to the successor's input list where it writes its output messages.

Tasks: Every task is mapped onto one ore more activities to ensure the compliment of the maximum part of the task's functionality. As we mentioned before, tasks are classified into 4 groups:

Constructors: Custom Tasks are converted into custom activities where user can insert custom code. For this purpose, the user can use some variables we predefined such as input messages to read the incoming messages and the output messages where it can write the resulting messages for the successors tasks. The splitter and its opposite task, the aggregator, were transformed using XPath to implement these tasks' functionalities.

An aggregator or splitter in our integration solution needs Xpath expressions, provided by user, to fragment an input message creating various output messages, or to aggregate the input messages creating a new one. When it is transformed into WF, a WF activity executes this XPath expression over the input messages and generates a unique output message.

Transformers: This type of tasks was implemented using XSL files that transform XML messages from one scheme to another. In the case of the translator, the user should indicate the XSL transformation that must be applied to get the new type of messages. In WF, this is an activity that reads the input message, uses the .Net API to apply XSL transformation over it, and writes the resulting message in the input buffer of the following activities. XSL transformations were also used for the Enricher and the Slimmer, whose functionalities are opposite to each other. In the case of the enricher, the XSL should query a database or some files, creating a new message with the old one and the new information, whereas the slimmer's XSL file creates a new message eliminating unnecessary data from the original message. Transformation of this group of tasks was simple using of the .Net API for XML transformation.

Routers: Filters seem like WF's If-Else activity, but implementing the conditions for this type of activity in WF is not so simple since they should be expressed in a complex XML structure. Another way to see this activity is as any other WF activity that before writing any message to the successor's input buffer, the activity checks if it satisfies the condition entered by user in the Filter task. Some variables are provided to the user so that conditions may be defined over input messages giving conditions more expressiveness. Another task in this group is the replicator, which was simple to implement using the XML API in the .Net to clone messages and place a copy for each successor. The Resequencer was not implemented because the message's model in our implementation doesn't have any identifier that can be used to order the incoming messages. The other two tasks of this group are the Distributor and the Merger. Due to some limitations that we discuss in the next section, the user must write his or her custom code in the case of the distributor, but he can make use of lists of messages where successor's messages are saved. This way, a user can introduce and check the conditions that should be satisfied before passing the messages to a successor task. In the case of the merger, the WF activity reads the XML messages from its input and creates an output message by gathering the input messages under a unique root. A replicator may have more than one output whereas a merger has more than one input. We simulated these tasks making use of the parallel activities of WF.

Interfacing: Neither does WF provide predefined activities, nor facilitates implementing the tasks in this group. These tasks were transformed into activities that call C# code where data bases, files and web services are accessed and information is read. Some scrappers were also implemented to be used inside our designed integration solutions.

5.2 Limitations

Similarities exist between workflows and integration solutions, but depending on the implementation, some features may be used, while other features are not available. In

our case, some of the characteristics of WF helped us, such as the XML language to define workflows, the workflows designer and the ability to execute our C# code inside our activities. Other functionalities were not implemented or were partially implemented.

In the case of the constructors, although WF provides an activity similar to the Timer called Delay Activity, this activity has a different purpose, since a Timer is a task that reads messages at regular time intervals, whereas the delay activity in WF causes a delay in execution of the workflow. Timers were not implemented due to the lack of a similar activity in WF and other limitations we mention in the next section.

In the routers group, a Distributor has a condition for every successor slot, and only writes the message in this slot if the condition is satisfied. Here slots can have conditions, buffer, and other properties. In a workflow, there exists a slight similar concept which is the arrow that interconnects activities. It is used to indicate that the activity connected to the origin is the predecessor of the task connected to the end of this arrow (target activity). This limitation made us implement slots in a different manner, so now it is the predecessor's responsibility to pass messages and to write them in the input buffer of the successors.

More serious limitations are the slot cardinalities. In our integration solution, a slot may have more than one input and more than one output, but for a workflow in WF, an activity has only one input, and a unique output. Tasks with more than one output (Replicator, Distributor, etc.) and tasks with more than one input like Merger, are implemented using parallel activities. In the first case, tasks with multiple outputs are always followed by a parallel activity where each successor activity occupies a branch of this parallel activity, while in the second case, this type of tasks always comes after a parallel activity where each branch of this parallel activity contains a preceding activity of the merger in our integration solution. An example can be seen in the second process of Figure 3, where a replicator is followed by a parallel activity while a merger is put just after a parallel activity.

The limitation that most affected this implementation, was the number of threads. In a workflow, it is supposed that a unique thread moves information and accomplishes tasks, while in our integration solution each task could be a thread. For example, while a task is reading from a database, another can be transforming the information read so far. The simulation of this feature, was done by creating an input directory, while a file system watcher monitors changes in this directory. When messages are created in this directory, an instance of the workflow, result of the transformation, is created and messages are injected to this workflow. The injected messages then flow inside the generated workflow where activities perform tasks' functionalities over these messages.

Using the WF implementation, some characteristics were lost, such as simultaneous tasks execution, multiple inputs, multiple outputs and the interfacing group which was not implemented at all, but other features and advantages were gained like WF services and the simplicity of the definition of workflows in WF. A workflow in WF is thought to implement the business logic inside a user application, helping the creation of Process-driven applications. WF doesn't offer activities to communicate with other applications (only with web services).

6 Conclusions and future work

During the last decades many monolithic, single-purpose applications were created for supporting companies' businesses. Part of these applications, nowadays called legacy systems, are still running in a distributed environment and so with other software packages purchased from third parties and new specially tailored applications, represent the software eco-system found in a large number of companies. The process of integrating applications inside or amongst different companies, known, respectively, as Enterprise Application Integration and Business to Business Integration, is still a very costly task. We believe a domain specific language and appropriate software tools to design platform independent models for integration solutions which can be programmatically transformed into a specific deployment technology, could help to reduce this cost.

In this paper, we introduced Guaraná, its editor and a transformer. We used DSL Tools technology provided by Microsoft to develop a software tool which implements the domain specific language Guaraná, allowing to visually design integration solutions using this language and also to deploy them to WF technology. In this first approximation to a programmatically transformation of platform independent models into executable code, we have also used WF's runtime environment to host and run the deployed integration solution. This experience has provided us sufficiently information to motivate us to improve our software tool by using the Model Driven Architecture approach to perform the transformation process from the platform independent model of an integration solution into executable code of a target technology.

References

1. A. Abouzahra, J. Bézivin, M. D. Del Fabro, and F. Jouault. A practical approach to bridging domain specific languages with UML profiles. In *Proceedings of the Best Practices for Model Driven Software Development at OOPSLA*, volume 5, 2005.
2. C. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan. Survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, 2006.
3. R. Corchuelo, R.Z. Frantz, and J. González. Una Comparación de ESBs desde la Perspectiva de la Integración de Aplicaciones. In *Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, 2008.
4. Microsoft Corporation. Windows Workflow Foundation Home, 2009.
5. R.Z. Frantz and R. Corchuelo. Integración de aplicaciones: Un lenguaje específico de dominio para el diseño de soluciones de integración. Technical report, Universidad de Sevilla, 2008.
6. R.Z. Frantz, R. Corchuelo, and J. González. Advances in a DSL for Application Integration. In *Proceedings of the Zoco08 Workshop*, pages 54–66, Gijón (España), 2008.
7. Bobby Woolf Gregor Hohpe. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.
8. Microsoft. Overview of domain-specific language tools, 2009.
9. Object Management Group (OMG). OMG EAI Profile Home, 2004.
10. Charles Plesums. Introduction to workflow. Technical report, Computer Sciences Corporation, Financial Services Group, 2002.
11. J. Weiss. Aligning relationships: Optimizing the value of strategic outsourcing. Technical report, IBM, 2005.

Reingeniería sobre Almacenes de Datos Seguros aplicando ADM

Carlos Blanco[1], Eduardo Fernández-Medina[1] y Juan Trujillo[2]

[1]Dep. de Tecnologías y Sistemas de Información – Escuela Superior de Informática
Grupo ALARCOS – Instituto de Tecnologías y Sistemas de Información
Universidad de Castilla-La Mancha
Paseo de la Universidad, 4. 13071. Ciudad Real, España
{Carlos.Blanco, Eduardo.Fdezmedina}@uclm.es
[2]Departamento de Lenguajes y Sistemas de Información. Facultad de Informática
Grupo de Investigación LUCENTIA
Universidad de Alicante
San Vicente s/n. 03690. Alicante, España
jtrujillo@dlsi.ua.es

Abstract. Los almacenes de datos (DWs) soportan el proceso de toma de decisiones manejando información histórica de la empresa, que debido a su vital importancia ha de ser protegida ante accesos no autorizados. Tradicionalmente el desarrollo de DWs o no soportaba la inclusión de medidas de seguridad o incorporaba algunas restricciones en la etapa final de desarrollo. De este modo nos encontramos con numerosos sistemas heredados que podrían beneficiarse de un proceso de reingeniería que redocumente el sistema y permita restablecer aspectos de seguridad a un mayor nivel de abstracción. Nuestra propuesta utiliza las ventajas del enfoque dirigido por modelos (MDA) para integrar aspectos de seguridad en todas las etapas del proceso de desarrollo del DW, proporcionando así un ahorro de tiempo y costo, a la vez que una mejor integración de la seguridad con el sistema. En este artículo mejoramos dicha arquitectura incorporando un proceso de modernización dirigido por la arquitectura (ADM) que partiendo de sistemas OLAP heredados obtiene modelos conceptuales que permiten el restablecimiento de medidas de seguridad y una posterior regeneración de código o migración a otras plataformas siguiendo el enfoque MDA.

1 Introducción

Los Almacenes de Datos (DWs) son repositorios de información histórica de negocio que es integrada desde diferentes fuentes de datos [1]. Esta información se suele organizar siguiendo un enfoque multidimensional compuesto por hechos (por ejemplo, la venta de un producto) y dimensiones relacionadas que clasifican la información por temas (por ejemplo, departamentos, ciudades o categorías de productos). La típica arquitectura de un almacén de datos está formada por varias capas: las fuentes de datos heterogéneas (Data Sources); los procesos ETL (Extraction / Transformation / Load) que extraen la información de las fuentes, la transforman y la cargan en el almacén; el repositorio del almacén que almacena los datos y

Actas JISBD 2009, pp.146-157, ISBN: 978-84-692-4211-7

representa la parte central de la arquitectura; y las herramientas finales que analizan los datos y que pueden ser sistemas gestores de bases datos (SGBD) o herramientas de procesamiento analítico en línea (OLAP).

El almacén de datos gestiona datos sensibles ya que maneja información de negocio usada para la toma de decisiones e información de carácter personal. Esta información ha de ser protegida frente a accesos no autorizados, siendo necesario integrar medidas de seguridad en todas las capas y operaciones del almacén, teniéndola en cuenta desde etapas tempranas de desarrollo como un requisito crítico hasta la implementación final en SGBD o herramientas OLAP [2].

Por otro lado, el enfoque de desarrollo dirigido por modelos [3] nos permite modelar el sistema utilizando distintos niveles de abstracción y transformaciones automáticas entre ellos. De este modo MDA proporciona modelos de negocio (CIM) que especifican los requisitos del sistema, modelos conceptuales (PIM) que representan el sistema independientemente de la plataforma y modelos lógicos (PSM) que se centran en una tecnología específica. La transformación automática entre modelos puede ser definida mediante la utilización de lenguajes como el propuesto por la OMG, Query / Views / Transformations (QVT) [4]. MDA también soporta procesos de modernización dirigidos por la arquitectura (ADM) [5] en los que se definen transformaciones en sentido ascendente de forma que partiendo de sistemas heredados se obtengan modelos a mayor nivel de abstracción que pueden ser más fácilmente analizados y mejorados con nuevos aspectos de seguridad. Posteriormente, y siguiendo el enfoque MDA, estos modelos pueden ser usados para regenerar el código fuente o migrar el sistema a diferentes plataformas.

Con el objetivo de desarrollar almacenes de datos considerando aspectos de confidencialidad en todo el proceso de desarrollo, hemos desarrollado una propuesta alineada con una arquitectura MDA [6] que utiliza varios modelos (CIM, PIM, PSM) ampliados con características de seguridad y transformaciones automáticas hacia implementaciones en SGBD o herramientas OLAP. Este artículo mejora la arquitectura incorporando un proceso de modernización (ADM) que permite redocumentar sistemas OLAP heredados y utilizar estos modelos más abstractos para detectar aspectos de seguridad y reimplementar o migrar el sistema de forma automática. Dicho proceso está centrado en la obtención de modelos a nivel conceptual (PIM) partiendo de modelos lógicos multidimensionales (PSM) correspondientes a implementaciones en herramientas OLAP.

La organización del artículo es la siguiente: la Sección 2 presenta las propuestas existentes en el desarrollo de DWs seguros incluyendo un resumen de nuestra arquitectura MDA; la Sección 3 presenta el proceso de modernización ADM definido en este trabajo y la Sección 4 un ejemplo del mismo; finalmente la Sección 5 muestra nuestras conclusiones y trabajo futuro.

2 Trabajo Relacionado

Dentro del desarrollo de Sistemas de Información Seguros existen varias contribuciones que consideran la seguridad a todos los niveles del proceso de desarrollo. Las más relevantes son UMLSec [7], que utiliza UML para definir y

evaluar especificaciones de seguridad usando una semántica formal, y Model Driven Security (MDS) [8], que incluye propiedades de seguridad en modelos a alto nivel utilizando una extensión de UML llamada SecureUML [9] y genera arquitecturas seguras de forma automática siguiendo el enfoque MDA.

Estas propuestas se centran en el desarrollo de Sistemas de Información en general y no tratan los problemas de seguridad específicos de DWs. En este ámbito la propuesta más interesante es la metodología de Priebe y Pernul [10] en la que analizan requisitos de seguridad, modelan el sistema a nivel conceptual utilizando ADAPTed UML y tratan con su implementación final en herramientas comerciales ocultando elementos multidimensionales. Sin embargo, en dicha metodología no se establecen las conexiones entre niveles que permitirían realizar transformaciones automáticas.

Nuestra propuesta para el desarrollo de DWs seguros utiliza el enfoque MDA para desarrollar una arquitectura [6] en la que se incluyen aspectos de confidencialidad en todas las etapas del proceso de desarrollo proporcionando modelos a distintos niveles de abstracción y transformaciones automáticas (Figura 1).

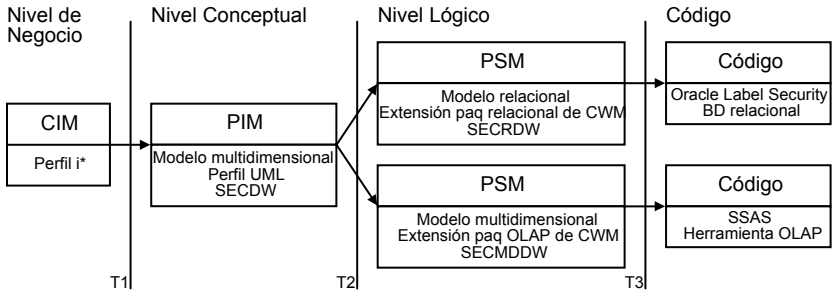


Figura 1. Desarrollo de DWs seguros mediante MDA

En primer lugar, un perfil UML [11] basado en el marco i* permite la especificación de requisitos de seguridad a nivel de negocio (CIM). A nivel conceptual (PIM), el sistema es definido utilizando un perfil UML específicamente creado para DWs, llamado SECDW [12], que ha sido complementado con un modelo de control de acceso y auditoría (ACA) [13]. Los aspectos de seguridad proporcionados por el modelo ACA son una triple clasificación de sujetos y objetos en niveles, roles y compartimentos de seguridad, y la definición de los siguientes tipos de reglas de seguridad: de asignación de información sensible, de autorización y de auditoría.

A continuación, se realiza el modelado a nivel lógico (PSM) dependiendo de la tecnología final utilizada, siguiendo principalmente un enfoque relacional (ROLAP), multidimensional (MOLAP) o híbrido (HOLAP). Dicha arquitectura incluye dos metamodelos a nivel lógico proporcionando así dos caminos distintos: uno relacional hacia SGBD y otro multidimensional hacia herramientas OLAP. Ambos metamodelos están basados en paquetes del Common Warehouse Metamodel [14], de forma que el metamodelo relacional, SECRDW [15], se basa en el paquete relacional permitiendo la definición de elementos relacionales como tablas o columnas, y el metamodelo multidimensional, SECMDDW [16], se basa en el paquete OLAP incluyendo

conceptos estructurales de cubos, dimensiones, medidas, etc. y permisos de seguridad relacionados con cubos, dimensiones y propiedades.

Siguiendo con la filosofía MDA, se han definido transformaciones automáticas entre modelos utilizando los estándares de la OMG (Figura 1): para la transformación T1 (CIM-PIM) se ha definido un proceso en SPEM que permite obtener los modelos conceptuales [17]; para las transformaciones T2 (PIM-PSM) que conducen hacia los dos modelos lógicos se han utilizado conjuntos de reglas modelo-a-modelo (M2M) implementadas con QVT [16, 18]; y finalmente, utilizando reglas de transformación modelo-a-texto (M2T) implementadas mediante MofScript, partiendo de los modelos PSM se ha obtenido código fuente para las herramientas Oracle Label Security (SGBD) y SQL Server Analysis Services (OLAP).

En cuanto a la ingeniería inversa, aunque esta ha sido ampliamente estudiada [19-22], no existen trabajos centrados en DWs y que incluyan aspectos de seguridad y que partiendo de sistemas heredados permitan obtener modelos de mayor nivel de abstracción que puedan ser mejorados y reimplementados de forma automática en la misma plataforma final o realizar migraciones hacia otras plataformas.

3 Proceso de Modernización de DWs Seguros

La incorporación de un proceso de modernización a la arquitectura proporciona varios beneficios. Principalmente se generan diagramas a mayor nivel de abstracción (PIM) que permiten re-documentar sistemas existentes e identificar necesidades de seguridad de una forma más fácil. A continuación, y utilizando las reglas de transformación definidas, se pueden obtener los diagramas lógicos (PSM) y el correspondiente código, permitiendo también la migración a diferentes tecnologías (utilizando otro PSM de destino, ROLAP, MOLAP, etc.) y herramientas finales. Debido a que la mayoría de los DWs son analizados mediante herramientas OLAP utilizando un enfoque multidimensional, en este trabajo nos centramos en definir un proceso de modernización sobre el camino multidimensional (Figura 1).

En una primera etapa, ha de extraerse el correspondiente modelo lógico multidimensional (PSM) de acuerdo al metamodelo SECMDDW, realizando un análisis estático del código fuente de la herramienta OLAP. El análisis estático [23] es un método de reingeniería basado en la generación de analizadores léxicos y sintácticos para una herramienta específica, que permiten analizar el código y aplicar transformaciones de texto a modelo para crear los elementos necesarios en el modelo PSM.

Una vez que el modelo lógico multidimensional (PSM) ha sido obtenido, es necesario aplicar un conjunto de reglas de transformación modelo a modelo (M2M) que permitan obtener el correspondiente modelo conceptual (PIM). En este trabajo se han definido varios conjuntos de reglas QVT que permiten realizar esta transformación de forma automática.

El metamodelo de partida, SECMDDW, presenta tres submodelos: configuración de seguridad, cubos y dimensiones. El de configuración de seguridad permite establecer la configuración del sistema en base a una política de control de acceso basada en roles (RBAC); y el de cubos y dimensiones, permiten representar a nivel

lógico aspectos estructurales (cubos, dimensiones, jerarquías, medidas, atributos, etc.) y permisos de seguridad relacionados con varios elementos multidimensionales (cubos, dimensiones, celdas y atributos).

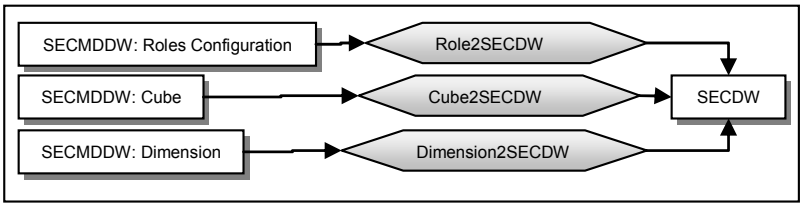


Figura 2. Transformación PSM a PIM

Tal y como se puede observar en la Figura 2, se han definido tres conjuntos de reglas QVT que partiendo de cada uno de estos modelos de origen, generan el modelo de destino de acuerdo a SECDW. Debido a que el modelo de destino (nivel conceptual) es más rico que el de partida, las reglas definidas han de ajustarse a este salto semántico. Un ejemplo de este salto semántico es la generación de la configuración de seguridad. Es decir, a nivel conceptual SECDW permite la definición de roles, niveles y compartimentos de seguridad pero a nivel lógico SECMDDW sólo utiliza roles. En el paso directo PIM a PSM hay una pérdida de semántica ya que se transforma toda esta información a roles, pero en paso inverso PSM a PIM sólo encontramos roles en el sistema heredado, lo cual nos obliga a crear sólo roles en el modelo de destino (PIM) y no utilizar los niveles y compartimentos de seguridad que también nos proporciona. La transformación "Role2SECDW" realiza esta transformación de configuración de seguridad mediante las reglas "RoleFiles2Package" y "Role2SRole", que crean un paquete con roles por cada rol detectado a nivel lógico. La Figura 3 muestra la regla "Role2SRole".

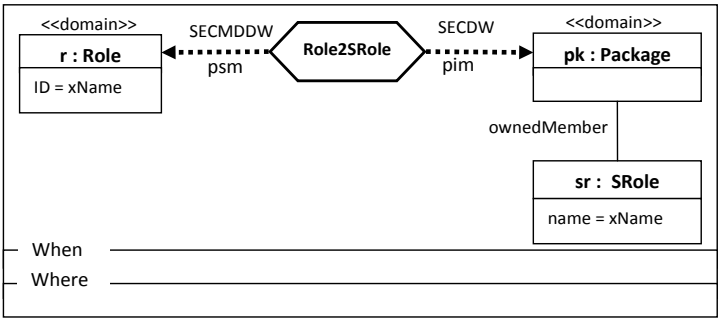


Figura 3. Regla Role2SRole

La transformación "Cube2SECDW" analiza los modelos lógicos de cubos y genera a nivel conceptual los aspectos estructurales y de seguridad necesarios. En primer lugar, la regla principal "CubeFiles2Package" crea el paquete para cubos y lanza las reglas estructurales encargadas de transformar los cubos en clases de tipo hecho (regla "Cube2SFact") (Figura 4), sus medidas asociadas en propiedades seguras (reglas

"Measure2SFA" y "Measure2Property"), y las dimensiones relacionados en dimensiones seguras (regla "Dimension2SDimension"). A continuación se lanzan las reglas de seguridad que se encargan de analizar los permisos de seguridad definidos a nivel de cubo o de celda y transformarlos a nivel conceptual en información de seguridad asociada a las clases de tipo hecho y propiedades correspondientes (reglas "CubePermission2SClass" y "CellPermission2SProperty").

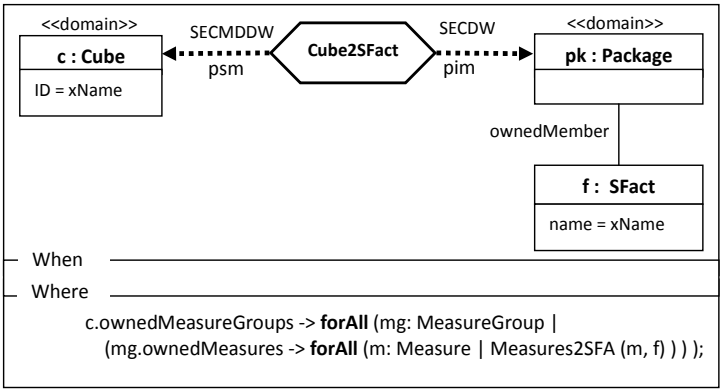


Figura 4. Regla Cube2SFact

Por último, la transformación "Dimension2SECDW" se centra en el modelo de partida de dimensiones. La regla principal "DimensionFiles2Package" analiza las dimensiones existentes y crea un paquete a nivel conceptual. Las reglas estructurales partiendo de las dimensiones, propiedades y jerarquías definidas, crean en el modelo conceptual las correspondientes clases del tipo dimensión (regla "Dimension2SDimension"), sus propiedades relacionadas (regla "attribute2SProperty") y las jerarquías y clases base (reglas "hierarchy2SBase" y "attribute2SBaseProperty"). A continuación, los permisos de seguridad establecidos sobre dimensiones y atributos son transformados en información de seguridad relacionada con las dimensiones y propiedades correspondientes (reglas "DimensionPermission2SClass" (Figura 5) y "AttributePermission2SProperty").

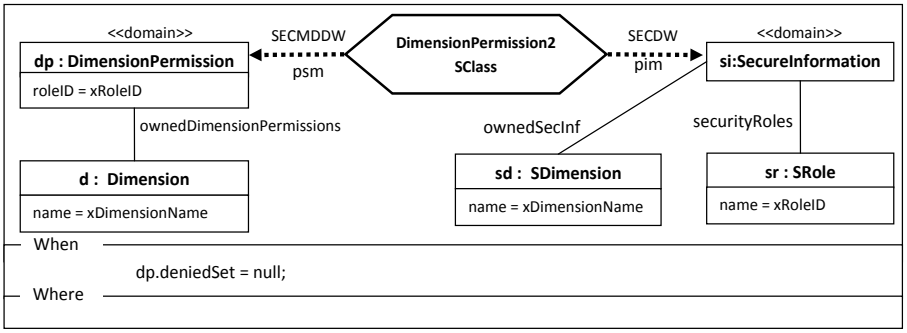


Figura 5. Regla DimensionPermission2SClass

A continuación en la Tabla 1, aparece el código correspondiente a varias de las reglas QVT desarrolladas.

Tabla 1. Código QVT para las transformaciones

```
transformation Role2SECDW (psm:SECMDDW, pim:SECDW) {
    key SECDW::SRole {rootPackage, name};
    top relation RoleFiles2Package { xName : String;
            checkonly domain psm rf:SECMDDW::SecurityConfiguration::RoleFiles {
                    name = xName };
            enforce domain pim pk:SECDW::Package { name = xName };
            where { rf.ownedRoles->forAll (r:SECMDDW::SecurityConfiguration::Role |
                    Role2SRole(r, pk)); }   }
    relation Role2SRole { xName : String;
            checkonly domain psm r:SECMDDW::SecurityConfiguration::Role { ID = xName };
            enforce domain pim pk: SECDW::Package{
                    ownedMember = sr : SECDW::SRole { name = xName } };   }}
transformation Cube2SECDW (psm:SECMDDW, pim:SECDW) {
    key SECDW::SFact {rootPackage, name};
    top relation CubeFiles2Package { xName : String;
            checkonly domain psm cf:SECMDDW::Cubes::CubeFiles { name = xName };
            enforce domain pim pk:SECDW::Package { name = xName };
            where { cf.ownedCubes->forAll (c:SECMDDW::Cubes::Cube | Cube2SFact(c, pk)); }   }
    relation Cube2SFact { xName : String;
            checkonly domain psm c:SECMDDW::Cubes::Cube { ID = xName };
            enforce domain pim pk: SECDW::Package {
                    ownedMember = f : SECDW::SFact { name = xName } };
            where { c.ownedMeasureGroups->forAll (mg:SECMDDW::Cubes::MeasureGroup |
            (mg.ownedMeasures->forAll (m:SECMDDW::Cubes::Measure | Measures2SFA(m, f))));}}
    relation Measures2SFA { xName : String;
            checkonly domain psm m:SECMDDW::Cubes::Measure { ID = xName };
            enforce domain pim f:SECDW::SFact {
                    attributes = sfa:SECDW::SFA { name = xName } };   }}
transformation Dimension2SECDW (psm:SECMDDW, pim:SECDW) {
    key SECDW::SDimension {rootPackage, name};
    key SECDW::SRole {rootPackage, name};
    top relation DimensionFiles2Package { xName : String;
            checkonly domain psm df:SECMDDW::Dimensions::DimensionFiles { name = xName };
            enforce domain pim pk:SECDW::Package { name = xName };
            where { df.ownedDimensions->forAll (d:SECMDDW::Dimensions::Dimension |
                     Dimension2SDimension(d, pk)); } }
    relation Dimension2SDimension { xName : String;
            checkonly domain psm d:SECMDDW::Dimensions::Dimension {ID = xName };
            enforce domain pim pk: SECDW::Package {
                    ownedMember = sd : SECDW::SDimension {
                    ownedSecInf = si : SECDW::SecureInformation {}, name = xName } };
            where { d.ownedDimensionPermissions->forAll
            (dp:SECMDDW::Dimensions::DimensionPermission  |
            (dp.deniedSet.oclIsUndefined()) implies (DimensionPermission2SClass (dp, si, pk)) ); }}
    relation DimensionPermission2SClass { xRoleID : String;
            checkonly domain psm dp:SECMDDW::Dimensions::DimensionPermission {
                    roleID = xRoleID };
            enforce domain pim  sd :SECDW::SecureInformation {
                    securityRoles = sr : SECDW::SRole { name = xRoleID } };
            enforce domain pim pk:SECDW::Package { ownedMember = sr : SECDW::SRole {} };
            when{ dp.deniedSet = ''; }  }}
```

4 Ejemplo

En esta sección se muestra el proceso de modernización (ADM) utilizando un pequeño ejemplo en el que se parte de los modelos lógicos multidimensionales (PSM) de un Almacén de Datos que gestiona las admisiones de un hospital. La Figura 6 muestra el modelo lógico con la configuración de seguridad del sistema en base a la definición de un conjunto de roles de seguridad; la Figura 7 muestra el modelo de cubos y la Figura 8 un modelo parcial de dimensiones en el que aparece la parte estructural y de seguridad de la dimensión Paciente. El modelo conceptual resultante tras aplicar las transformaciones definidas en este artículo aparece en la Figura 9.

En primer lugar la transformación **Role2SECDW** se encarga de obtener los roles presentes en el modelo PSM de configuración de seguridad y generar por cada uno un rol en el modelo de destino (PIM). Se puede observar como en el modelo lógico (Figura 6) aparecen unos roles que representan una jerarquía de roles ("EmpleadoDeHospital", "Sanitario", "NoSanitario", etc.) y otros que representan una serie de niveles de seguridad ("SLAltoSecreto", "SLSecreto", etc.). Esto es un ejemplo de la pérdida de semántica que comentábamos anteriormente. En este caso aunque por la nomenclatura podríamos deducir que unos roles representan niveles de seguridad y otros no, en cualquier sistema heredado no podríamos conocer esta información ni la relación existente entre los roles para poder establecer las correspondientes relaciones de la jerarquía. De este modo las transformaciones definidas se encargan de transformar cada rol del nivel lógico a un rol a nivel conceptual, sin utilizar los demás mecanismos ofrecidos por el metamodelo conceptual (compartimentos y niveles de seguridad).

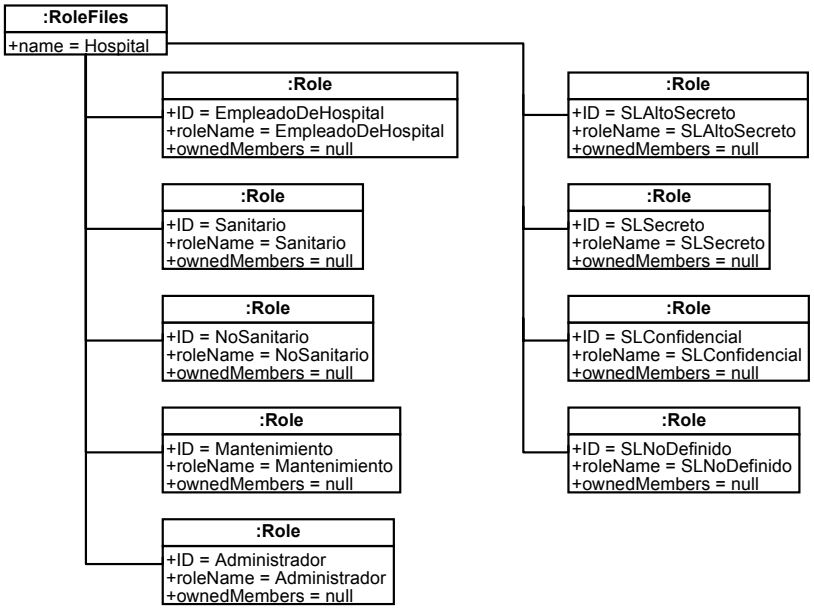


Figura 6. Modelo lógico multidimensional (PSM): configuración de seguridad

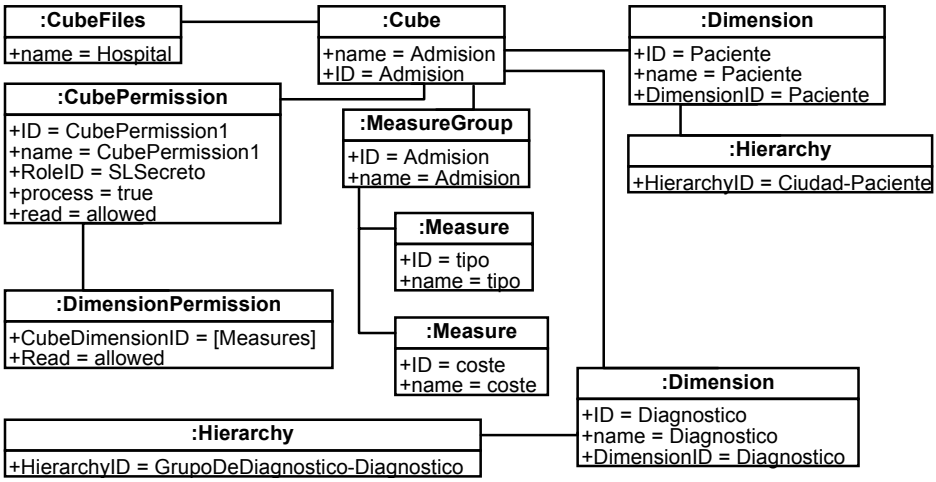


Figura 7. Modelo lógico multidimensional (PSM): cubos

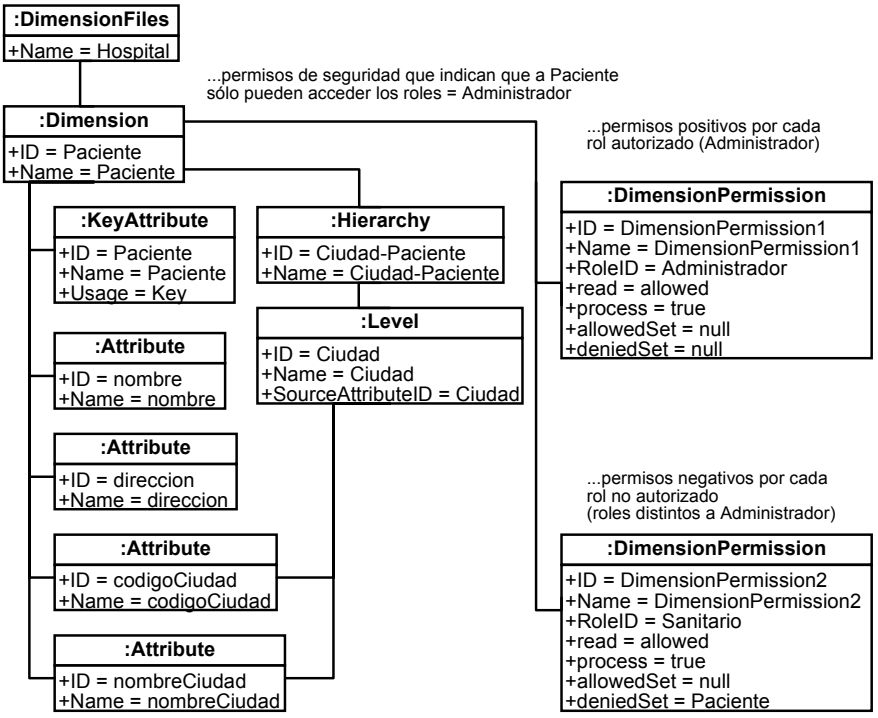


Figura 8. Modelo lógico multidimensional (PSM): dimensión Paciente

A continuación, la transformación **Cube2SECDW** parte del modelo de cubos (Figura 7) y crea en el modelo conceptual (Figura 9) la clase hecho "Admision", sus medidas asociadas ("tipo" y "coste") y sus relaciones con dimensiones ("Paciente" y "Ciudad") y jerarquías ("Diagnostico" y "GrupoDeDiagnostico"). En cuanto a los

aspectos de seguridad, ha sido definido a nivel lógico un permiso de cubo indicando que sólo pueden acceder a él usuarios con rol "SLSecreto". Dicha restricción de seguridad es transformada a nivel conceptual como información de seguridad asociada a la clase hecho "Admision".

Finalmente, la transformación **Dimension2SECDW** parte del modelo de dimensiones (la Figura 8 muestra parcialmente este modelo) y crea en el modelo conceptual de destino las partes estructurales relacionadas con las dimensiones ("Paciente" y "Diagnostico"), sus atributos, y jerarquías, mediante la inclusión de clases base relacionadas y sus atributos ("Ciudad" y "GrupoDeDiagnostico"). En el modelo lógico de la dimensión "Paciente" han sido definidos un conjunto de permisos de dimensión que autorizan sólo a los usuarios con rol "Administrador" el acceso a la dimensión. Estos permisos son transformados a nivel conceptual como estereotipos con información de seguridad asociada a la clase "Paciente", indicando que solo el rol "Administrador" puede acceder a ella.

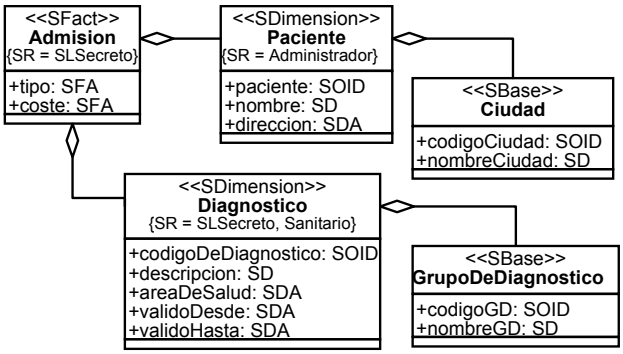


Figura 9. Modelo conceptual (PIM)

5 Conclusiones

Nuestro trabajo se centra en la definición de sistemas de información seguros incorporando aspectos de seguridad desde fases tempranas del proceso de desarrollo. En este sentido, hemos tratado con el desarrollo de Almacenes de Datos seguros mediante la definición de varios modelos mejorados con aspectos de seguridad, que permiten representar el almacén a diferentes niveles de abstracción. Dicha propuesta ha sido alineada con una arquitectura MDA, incluyendo así los niveles CIM, PIM y PSM, y transformaciones automáticas que nos permiten generar código final para SGBD, siguiendo un enfoque relacional o para herramientas OLAP, siguiendo un enfoque multidimensional.

Debido a las ventajas que proporciona la reingeniería, en este trabajo se ha abordado la incorporación de un proceso de modernización (ADM) sobre el camino multidimensional que permite obtener modelos conceptuales (PIM) partiendo de sistemas OLAP. De este modo nuestra arquitectura MDA soporta la redocumentación de sistemas heredados y permite utilizar estos modelos de nivel de abstracción más alto para detectar fallos de seguridad y realizar más fácilmente cambios que

posteriormente se transformen automáticamente hacia la implementación final. Del mismo modo y siguiendo la filosofía MDA y las transformaciones incorporadas en nuestra arquitectura, una vez obtenido el modelo conceptual podemos migrar el sistema hacia otra herramienta o tecnología.

Nuestros trabajos futuros se centran por un lado en el tratamiento del problema de la inferencia mediante la incorporación de modelos de seguridad dinámicos que complementen los modelos existentes y por otro lado en la incorporación de nuevos modelos PSM (como HOLAP o XOLAP) y herramientas finales (como Pentaho) a la arquitectura, permitiendo así un mayor soporte para las tecnologías y herramientas existentes.

Agradecimientos. Esta investigación es parte de los proyectos ESFINGE (TIN2006-15175-C05-05) del Ministerio de Educación y Ciencia, QUASIMODO (PAC08-0157-0668) de la Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha (JCCM) y el FEDER, SISTEMAS (PII2190-0150-3135) de la Consejería de Ciencia y Tecnología de la JCCM, MITOS (TC20091098) de la Universidad de Castilla-La Mancha y MEDUSAS (IDI-20090557), financiado por el centro para el Desarrollo Tecnológico Industrial (CDTI) del Ministerio de Ciencia e Innovación.

Referencias

1. Inmon, H., *Building the Data Warehouse*. Third Edition ed. 2002, USA: John Wiley & Sons.
2. Thuraisingham, B., M. Kantarcioglu, y S. Iyer, *Extended RBAC-based design and implementation for a secure data warehouse.* International Journal of Business Intelligence and Data Mining (IJBIDM), 2007. **2**(4): p. 367-382.
3. MDA, O.M.G., *Model Driven Architecture Guide.* 2003.
4. OMG, *MOF QVT final adopted specification.* 2005.
5. OMG, *ADM Glossary of Definitions and Terms.* http://adm.omg.org/ADM_Glossary_Spreadsheet_pdf.pdf. 2006, OMG. p. 34.
6. Fernández-Medina, E., J. Trujillo, y M. Piattini, *Model Driven Multidimensional Modeling of Secure Data Warehouses.* European Journal of Information Systems, 2007. **16**: p. 374-389.
7. Jürjens, J., *Secure Systems Development with UML*. 2004: Springer-Verlag.
8. Basin, D., J. Doser, y T. Lodderstedt, *Model Driven Security: from UML Models to Access Control Infrastructures.* ACM Transactions on Software Engineering and Methodology, 2006. **15**(1): p. 39-91.
9. Lodderstedt, T., D. Basin, y J. Doser. *SecureUML: A UML-based modeling language for model-driven security*. in *UML 2002. The Unified Modeling Language. Model Engineering, Languages Concepts, and Tools. 5th International Conference*. 2002. Dresden, Germany: Springer.
10. Priebe, T. y G. Pernul. *A Pragmatic Approach to Conceptual Modeling of OLAP Security*. in *20th International Conference on Conceptual Modeling (ER 2001)*. 2001. Yokohama, Japan: Springer-Verlag.
11. Trujillo, J., E. Soler, E. Fernández-Medina, y M. Piattini, *A UML 2.0 Profile to define Security Requirements for DataWarehouses.* Computer Standard and Interfaces, 2008. **in Press**.
12. Fernández-Medina, E., J. Trujillo, R. Villarroel, y M. Piattini, *Developing secure data warehouses with a UML extension.* Information Systems, 2007. **32**(6): p. 826-856.

13. Fernández-Medina, E., J. Trujillo, R. Villarroel, y M. Piattini, *Access Control and Audit Model for the Multidimensional Modeling of Data Warehouses.* Decision Support Systems, 2006. **42**: p. 1270-1289.
14. CWM, O.M.G., *Common Warehouse Metamodel (CWM)*. 2003.
15. Soler, E., J. Trujillo, E. Fernández-Medina, y M. Piattini, *Building a secure star schema in data warehouses by an extension of the relational package from CWM.* Computer Standard and Interfaces, 2008. **30**(6): p. 341-350.
16. Blanco, C., I. García-Rodríguez de Guzmán, D. G.Rosado, E. Fernández-Medina, y J. Trujillo, *Applying QVT in order to implement Secure Data Warehouses in SQL Server Analysis Services.* Journal of Research and Practice in Information Technology, 2009. **41**(2): p. 119-138.
17. Trujillo, J., E. Soler, E. Fernández-Medina, y M. Piattini, *An Engineering Process for Developing Secure Data Warehouses.* Information and Software Technology, 2008. **in Press**.
18. Soler, E., J. Trujillo, E. Fernández-Medina, y M. Piattini. *A Set of QVT relations to Transform PIM to PSM in the Design of Secure Data Warehouses*. in *IEEE International Symposium on Frontiers on Availability, Reliability and Security (FARES 2007)*. 2007. Viena, Austria.
19. Hainaut, J.-L., V. Englebert, J. Henrard, J.-M. Hick, y D. Roland1, *Database reverse engineering: From requirements to CARE tools*, in *Applied Categorical Structures. SpringerLink*. 2004.
20. Aiken, P.H., *Reverse engineering of data.* IBM Syst. J., 1998. **37**(2): p. 246-269.
21. Cohen, Y. y Y.A. Feldman, *Automatic high-quality reengineering of database programs by abstraction, transformation and reimplementation.* ACM Trans. Softw. Eng. Methodol., 2003. **12**(3): p. 285-316.
22. Blaha, M. *A Retrospective on Industrial Database Reverse Engineering Projects-Part 1*. in *Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01)*. 2001. Suttgart, Germany: IEEE Computer Society.
23. Canfora, G. y M.D. Penta, *New Frontiers of Reverse Engineering*. 2007 Future of Software Engineering. 2007: IEEE Computer Society. 326-341.

Wires* : A Tool for Orchestrating Model Transformations

J.E. Rivera, D. Ruiz-González, F. López-Romero, and J. Bautista

GISUM/Atenea Research Group. Universidad de Málaga (Spain).
{rivera,daniruiz,fernando,jmbautista}@lcc.uma.es

1 Introduction

Model-Driven Engineering (MDE) advocates the use of models as the key artifacts in all phases of development, from system specification and analysis, to design and implementation. Each model usually addresses one concern, independently from the rest of the issues involved in the construction of the system. The implementation of a system is then obtained by applying a set of transformations to the different models defined for it. These model transformations are usually interconnected, i.e., the output models of some transformations are the input models of others.

Model Transformation Orchestration (MTO) aims at supporting the construction of complex model transformations from other transformations already defined. MTO is especially relevant in the context of Global Model Management, in which megamodels are kinds of registries for resources available from a given model-driven platform, recording all accessible entities like models, metamodels, transformations, tools, and the various relations between them. In this setting, we expect that model transformations become available off-the-shelf and that can be reused to build new tools, processes and, of course, new transformations.

Wires* [1] is a Domain Specific Language that enables the high-level orchestration of model transformations. It provides a visual notation for defining chains of model transformation in a modular and compositional manner, and it is supported by a graphical framework and an execution engine that loads the appropriate models and execute the transformations along the pre-defined path.

2 The Wires* Language

Wires* assumes a data-flow process, in which a set of input models (conforming to their corresponding metamodels) are processed by a chain of ATL transformations until a set of output models is produced. The chain is composed of transformations, which act as processing nodes. Parameters represent the consumed and produced data by transformations. Transformations are wired together by directed connectors (called *Dataflows* or, simply, *Wires*) that indicate how the outputs of the transformations are linked to the inputs of the next ones. Fig. 1 shows a simple example, where an input model m1 is transformed by two transformations in a row to produce an output model m3.

The following paragraphs describe the main concepts of the language with more detail.

Actas JISBD 2009, pp.158-161, ISBN: 978-84-692-4211-7

Fig. 1. A simple chain of 2 model transformations.

Models. In our approach we consider that models and transformations are course-grained building blocks of the MDE process. Models are typed by the initial metamodel they conform to. Models without incoming *Dataflows* represent the input models of the process (i.e., the models that are initially loaded), while models with incoming *Dataflows* represent output models (i.e., models that are saved). The outputs of transformations that are not connected to a model element (see, e.g., the output of t1 in Fig. 1) are not saved at the end of the process. Finally, primitive data types are also supported.

Transformations. We contemplate the three different kinds of units defined by ATL: modules, libraries and queries. The instances of these units are separated from their corresponding specifications, on which we specify the data file where they are stored, their input and output formal parameters, as well as the auxiliary libraries they make use of. ATL modules may have multiple input and multiple output models. Queries have one single output, which is a primitive type value; one common use of a query is the generation of a textual output (encoded in a string value), which can be stored in a text file.

In addition to *Atomic* transformations, which represent basic ATL transformations, we also allow the definition of *Composite* transformations, which represent chains of transformations which can be later used as a single (atomic) transformation, i.e., as building blocks to specify further transformations. Fig. 2 shows an example of the specification of a composite model transformation, CT1, made of t1 followed by t2.

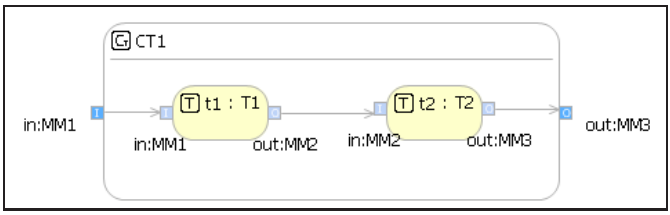


Fig. 2. The specification of a *Composite* transformation

We have also identified two special kinds of atomic transformations: *Identity* and *Generic*. Identity transformations are model transformations that do not alter the data which flow through them. (they are very useful, e.g., for defining conditional compositions of transformations [1]). Generic transformation are those used to load and execute ATL transformations produced by high-order transformations. They have a special input parameter (*typeParam*) that should conform to the ATL metamodel.

Fig. 3 depicts an example of the specification of a generic transformation. It shows a high-order transformation HOT that produces a transformation outTransf, which is passed as input to the generic transformation gt. Then, gt builds the ATL code corresponding to the model outTransf, compiles it and executes it on the input model m1 to produce model m3.

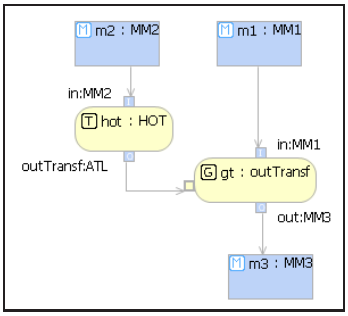


Fig. 3. The specification of a *Generic* transformation

Control Structures. Of course, not all chains of model transformations follow linear paths; for instance, models can have several outgoing *Dataflows*. They enable implementing parallel composition of transformations. Furthermore, the Wires* language also includes decision nodes whose input parameters are the outputs of queries (i.e., primitive data types), and that contain an OCL expression that will be evaluated to decide which one of its two branches (the true branch or the false branch) is enabled.

Fig. 4 (a) shows an example of a conditional composition of transformations, using queries and decision nodes. Assuming that we have two different refactoring algorithms (Refactor1 and Refactor2, implemented by two model transformations), the composition applies one or the other to an input model m1 depending on the deep inheritance tree (implemented by query DIT) of m1.

Loops can be simply expressed by using decision nodes and *Dataflows* to connect one of their branches to the beginning of the loop. For instance, Fig. 4 (b) shows the Wires* specification of a process that, given an endogenous model transformation b and an initial model in, iteratively applies b until it reaches a fixed point (i.e., its input model coincides with its output model). Equals is a query that checks whether its two input models are equal or not, returning a Boolean value with the result.

Notice that if b is an endogenous transformation defined by a set of rules, which are used to specify the behavior of metamodel MM1 (see [2]), and provided that ATL supports this kind of in-place semantics, then the process depicted in Fig. 4 (b) implements the simulation of the behavior of the system, from an initial state defined by model in. If required, one additional model transformation could take the intermediate output models m1' and visualize them, hence enabling the visualization of the successive states of the system during the simulation, and therefore producing a motion picture show.

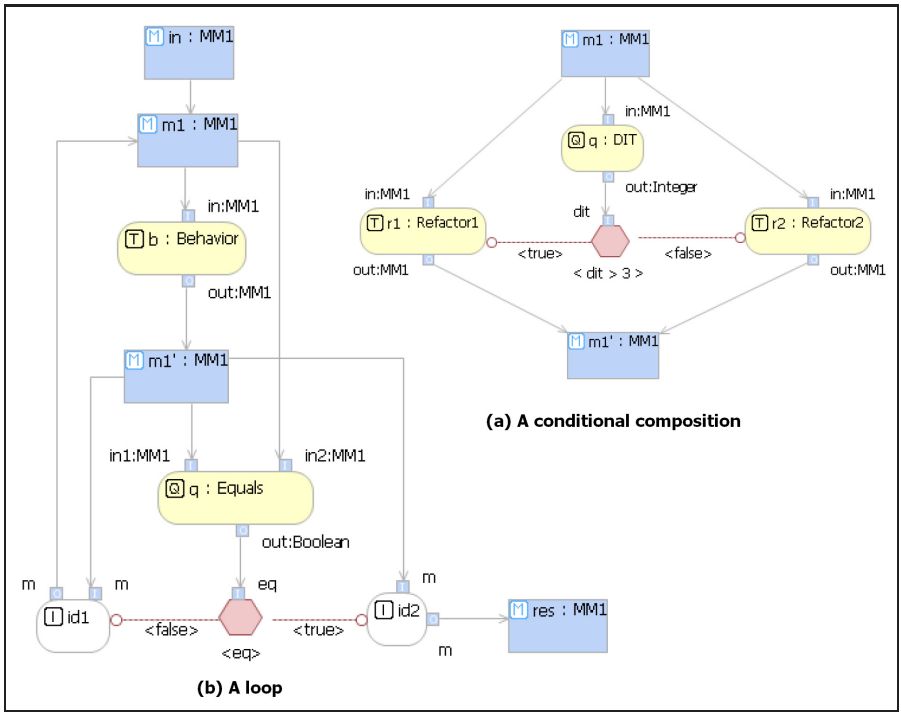


Fig. 4. Control Structures in Wires*

Acknowledgements. This work has been supported by Spanish Research Projects PET2006-0682-00, P07-TIC-03184 and TIN2008-03107.

References

1. Rivera, J.E., Ruiz-González, D., López-Romero, F., Bautista, J., Vallecillo, A.: Orchestrating ATL model transformations. In: To appear in Proc. of MtATL 2009
2. Rivera, J.E., Guerra, E., de Lara, J., Vallecillo, A.: Analyzing rule-based behavioral semantics of visual modeling languages with Maude. In: Proc. of SLE'08. LNCS, Toulouse, France, Springer (October 2008)

Gra2MoL: Una Herramienta para la Extracción de Modelos en Modernización de Software

Javier Luis Cánovas Izquierdo and Jesús García Molina

Grupo Modelum. Universidad de Murcia {jlcanovas,jmolina}@um.es

1 Introducción

La Modernización Dirigida por Modelos ha emergido recientemente como una nueva área dedicada a la automatización basada en modelos de procesos de modernización o evolución de software. Así, el OMG ha propuesto varios estándares de modernización dentro de la iniciativa ADM [1], como KDM [2]. En los próximos años será necesario un gran esfuerzo para encontrar técnicas y métodos para esta nueva área y será crucial disponer de herramientas que los soporten.

En general, el proceso de modernización asociado a un escenario de migración de plataformas consta de tres pasos principales: extracción de modelos a partir de los artefactos del sistema existente (ingeniería inversa), transformación de los modelos del sistema existente para generar modelos del sistema destino (rediseño) y generación de los artefactos del sistema destino (ingeniería directa). El paso dedicado a la extracción es realizado normalmente por soluciones *ad-hoc* basadas en la creación de analizadores específicos para realizar transformaciones texto-a-modelo. Con la finalidad de facilitar este paso, el grupo Modelum ha desarrollado un lenguaje de transformación texto-a-modelo denominado Gra2MoL [3, 4] que permite extraer modelos a partir de los ficheros de código fuente. De esta forma, una vez se dispone de los modelos del sistema, pueden ser llevadas a cabo las labores de reingeniería y generación del nuevo sistema aplicando transformaciones modelo-a-modelo o modelo-a-texto, completando el proceso de modernización.

2 Gra2MoL

Gra2MoL es un lenguaje basado en reglas que define un proceso de extracción de modelos como una transformación texto-a-modelo donde se especifican explícitamente las correspondencias entre los elementos de la gramática y los del metamodelo. La estructura de las reglas es similar a la considerada en lenguajes de transformación de modelos como ATL o RubyTL, con dos diferencias importantes: (1) el elemento origen de una regla es un elemento de la gramática en vez de un elemento del metamodelo y (2) la navegación es realizada por medio de un lenguaje de consultas adaptado para recorrer árboles de análisis, en vez de utilizar OCL o un lenguaje similar.

En Gra2MoL, cada regla especifica la correspondencia entre un elemento de la gramática y un elemento del metamodelo destino. Una regla tiene cuatro secciones: *from*, *to*, *queries* y *mappings*. La sección *from* especifica el símbolo no terminal de la gramática origen y declara una variable que referenciará al nodo del árbol de análisis

Actas JISBD 2009, pp.162-165, ISBN: 978-84-692-4211-7

cuando la regla sea ejecutada. Esta variable puede ser utilizada por cualquier expresión dentro de la regla. Además, esta sección puede incluir opcionalmente un filtro con una condición para seleccionar los nodos que pueden ejecutar la regla. La sección *to* especifica la metaclase del elemento del metamodelo destino. La sección de *queries* contiene un conjunto de expresiones de consulta que permiten extraer información del árbol de análisis. El resultado de estas consultas puede ser utilizado en las asignaciones de la sección de *mappings*. Finalmente, la sección de *mappings* contiene un conjunto de *bindings* para asignar valores a las propiedades de los elementos del modelo destino en base a las variables definidas en la sección de consultas.

Gra2MoL incorpora un potente lenguaje de consultas inspirado en XPath que permite recorrer el árbol de análisis de forma eficiente y extraer información dispersa a lo largo del código. Una consulta consiste en una secuencia de operaciones que incluyen: un operador, un tipo de nodo y expresiones opcionales de acceso y de filtro. Hay tres tipos de operadores de consulta: /, // y ///. El operador / devuelve los hijos inmediatos de un nodo y es similar al uso de la notación punto. Por otra parte, los operadores // y /// permiten la navegación de todos los hijos del nodo (directos e indirectos) recuperando todos los nodos de un tipo dado. Estos dos operadores permiten ignorar nodos superfluos intermedios, facilitando la definición de la consulta, dado que se especifica qué tipo de nodos deben ser encontrados pero no cómo alcanzarlos. Dado que una consulta puede devolver uno o más subárboles, el operador # es utilizado en una de las operaciones de consulta para indicar el tipo de los nodos resultado. Las operaciones de consulta pueden incluir una expresión de filtro opcional tal que solo aquellos nodos que satisfagan dicha expresión serán seleccionados. También pueden incluir opcionalmente una expresión de acceso, la cual es utilizada para acceder a nodos hermanos del árbol de análisis por medio de índices.

La ejecución de una transformación Gra2MoL está dirigida por los *bindings* contenidos en la sección de *mappings*. Estos *bindings* son un tipo especial de asignación similar a la utilizada en lenguajes de transformación de modelos como RubyTL y ATL. La parte izquierda de la asignación debe ser una propiedad de la metaclase destino, mientras que la parte derecha puede ser un valor literal, un identificador de consulta o una expresión. En el primer caso, el valor se asigna directamente. En el segundo caso, la consulta es aplicada y se ejecuta aquella regla cuyos tipos de la sección *from* y *to* conforman con los tipos de la parte derecha e izquierda, respectivamente. Las reglas de conformancia son explicadas en [4]. Finalmente, en el tercer caso, la expresión es evaluada y si el resultado es un terminal, el valor se asigna directamente; si el resultado es un nodo, se ejecutará la regla asociada al *binding*.

3 Ejemplo

A continuación presentamos un ejemplo de Gra2MoL extraído de un proyecto de migración entre dos plataformas Java: Struts y JSF. Para favorecer la interoperabilidad se utilizó KDM como metamodelo para representar el código Java. Por cuestiones de espacio, en esta sección solamente se muestra un extracto de la definición de transformación Gra2MoL que extrae modelos a partir de código Java [1].

[1] La definición de transformación completa puede ser descargada de http://modelum.es/gra2mol

```
rule 'mapCodeModel'
  from compilationUnit cu
  to code::CodeModel
  queries
    clases : /cu/#normalClassDecl;
  mappings
    name = "codeModel";
    codeElement = classes;
end_rule

rule 'mapClassUnit'
  from normalClassDecl nc
  to code::ClassUnit
  queries
    elements : /nc/#classBodyDecl;
  mappings
    name = nc.Id;
    codeElement = elements;
end_rule

rule 'mapField'
  from classBodyDecl//fieldDecl
  to code::MemberUnit
  queries
    ...
  mappings
    ...
end_rule

rule 'mapMethod'
  from classBodyDecl//memDecl{Id.exists}
  to code::MethodUnit
  queries
    ...
  mappings
    ...
end_rule
```

Fig. 1. Fragmento de una transformación Gra2MoL para la extracción de modelos KDM a partir de código fuente Java.

La transformación arranca con la ejecución de la primera regla de la definición, llamada `mapCodeModel`, de modo que se crea una instancia de la metaclase `CodeModel`. Mientras que el primer *binding* de esta regla asigna un valor al atributo `name`, el segundo provoca la ejecución de la regla `mapClassUnit`, ya que conforma con las secciones *from* y *to* de dicha regla. La regla `mapClassUnit` tiene también un primer *binding* que asigna un valor al atributo `name` y el segundo binding provoca la ejecución de las reglas `mapField` y `mapMethod` dependiendo de los nodos resultado de la consulta `elements`, los cuales deben cumplir la condición del filtro *from* de estas reglas.

4 Descripción de la herramienta

La herramienta desarrollada para soportar Gra2MoL dispone de dos tareas Ant encargadas de realizar las labores principales de una transformación. Una de ellas se encarga de enriquecer la gramática del código fuente para que contemple la generación de un árbol de análisis a partir del código. A partir de la gramática enriquecida se obtiene un analizador con ANTLR. La segunda tarea se encarga de ejecutar la definición de la transformación Gra2MoL. Esta tarea tiene cuatro entradas: el analizador creado por la tarea anterior, el código fuente, la definición de la transformación y el metamodelo destino. Como resultado de esta tarea, se obtienen los modelos extraídos del código fuente.

Aunque la herramienta ha sido inicialmente diseñada para ser utilizada de forma totalmente independiente de la plataforma Eclipse, en la actualidad se está llevando una labor de integración en la plataforma AGE [5]. AGE (*Agile Generative Environment*) es un entorno desarrollado también por el grupo Modelum cuyo principal objetivo es la experimentación con diferentes características de lenguajes de transformación, al mismo tiempo que es posible realizar desarrollos en contextos reales. Actualmente se dipone de un editor específico para las transformaciones Gra2MoL con resaltado de sintaxis, vista *outline* y autocompletado para la definición de las consultas, facilitando al desarrollador navegar por la gramática. La Figura 2 muestra una captura del editor y la vista *outline*.

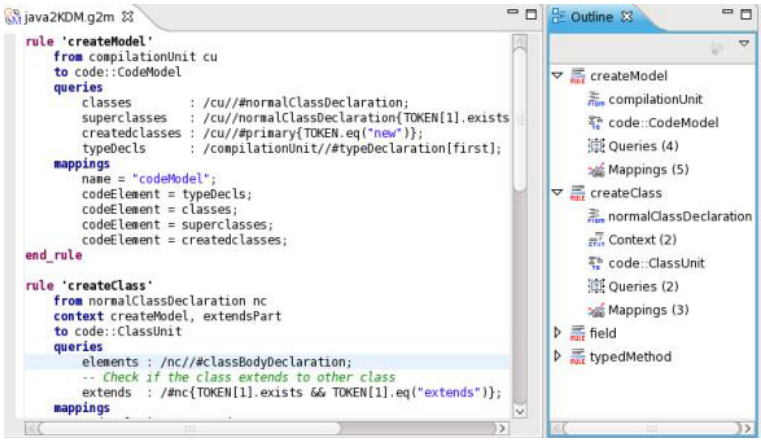


Fig. 2. Captura del entorno de desarrollo de la herramienta

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto 08797/PI/08 de la Fundación Séneca. Javier Luis Cánovas Izquierdo dispone de una beca FPI de la Fundación Séneca.

References

1. ADM. http://adm.omg.org
2. ADM Task Force: Knowledge discovery meta-model (kdm). OMG (2007).
3. J. Cánovas, O. Sánchez, J. S. Cuadrado, J. G. Molina, “DSLs para la extracción de modelos en modernización”. V Taller sobre Desarrollo de Software Dirigido por Modelos enmarcado en las XIII Jornadas de Ingeniera del Software y Bases de Datos (2008).
4. J. Cánovas, J. G. Molina, “A domain specific language for extracting models in software modernization”. V European Conference on Model-Driven Architecture Foundations and Applications, ECMDA09 (2009).
5. J. S. Cuadrado, J. G. Molina, “AGE (Agile Generative Environment)”. Sesión de demostraciones de herramientas en XII Jornadas de Ingeniería del Software y Bases de Datos (2007).

MoteGen: Una herramienta para el desarrollo de aplicaciones para redes de sensores

Fernando Losilla, Pedro Sánchez, Bárbara Alvarez, Diego Alonso,

Universidad Politécnica de Cartagena, Depto. TIC, 30202 Cartagena, España
{Fernando.Losilla, Pedro.Sanchez, balvarez, Diego.Alonso}@upct.es

Abstract. El desarrollo de software para redes de sensores inalámbricas implica contar con ciertos conocimientos sobre programación que pueden resultar complejos de adquirir. En este sentido el uso de un Lenguaje Específico de Domino puede simplificar el proceso de desarrollo, reduciendo tanto el tiempo necesario para construir aplicaciones como el empleado en adquirir los conocimientos necesarios. Un lenguaje de este tipo puede además servir como punto de partida de un enfoque de Desarrollo de Software Dirigido por Modelos, con la finalidad de obtener código ejecutable a partir de las descripciones realizadas con él. La herramienta *MoteGen* presentada sigue este enfoque, contando con un Lenguaje Específico de Dominio para redes de sensores inalámbricas a partir del que, mediante una serie de transformaciones sucesivas, obtiene, de forma totalmente automatizada, código ejecutable en el lenguaje de programación nesC usado por el sistema operativo TinyOS.

Keywords: Desarrollo de Software Dirigido por Modelos, Redes de sensores.

1 Introducción

El desarrollo de software para redes de sensores inalámbricas (*Wireless Sensor Networks*, WSN) es un proceso relativamente complejo que exige un nivel mínimo de conocimientos y habilidades a los desarrolladores de aplicaciones. A pesar de que se han producido muchos avances en las distintas áreas tecnológicas relacionadas con las redes de sensores, pocos de ellos se han centrado en la simplificación del proceso de desarrollo. Hasta no hace mucho lo más cercano a esto que se había hecho era la creación de nuevas tecnologías *middleware* que facilitan la construcción de aplicaciones complejas, pero que aún requieren del uso de lenguajes textuales que no resultan fáciles de usar.

La herramienta aquí presentada hace uso de un Lenguaje Específico de Dominio (LED)[1] gráfico para redes de sensores el cual está integrado en un proceso de Desarrollo de Software Dirigido por Modelos (DSDM) [2]. El objetivo que persigue es permitir la construcción de aplicaciones WSN de forma más sencilla e intuitiva, obteniendo código ejecutable de forma automática a partir de las descripciones del LED. Para tal fin se ha hecho uso de los niveles de abstracción definidos por el estándar MDA[3] así como de entornos de desarrollo y lenguajes de libre distribución.

En los siguientes apartados se describen superficialmente las redes de sensores y los fundamentos en los que se basa la herramienta desarrollada. A continuación se exponen los trabajos futuros que se van a acometer sobre la herramienta.

2 Redes de Sensores Inalámbricas

Las redes de sensores inalámbricas (*Wireless Sensor Networks*, WSN) [4] constituyen una tecnología de adquisición de datos que recientemente está atrayendo gran interés gracias a sus posibilidades, siendo aplicadas en numerosos ámbitos científicos e industriales para la realización de estudios y control de procesos. El uso de comunicaciones inalámbricas permite que los dispositivos sensores que las forman sean emplazados fácilmente sobre el terreno. Esto, unido al bajo coste de cada uno de ellos, permite el despliegue de un gran número de dispositivos que actúan como nodos de la red.

Para evitar tener que trabajar directamente con los recursos hardware de los elementos que forman las redes se recurre al uso de sistemas operativos. Éstos son específicos para las redes de sensores, manejando aspectos de gran importancia en esta tecnología como el ahorro de energía y las limitaciones de los recursos hardware. De entre todos los sistemas operativos TinyOS [5] es el más extendido. Este sistema operativo cuenta con un lenguaje de programación, nesC [6], que es una extensión de C orientada a componentes, lo que facilita en gran medida la reutilización de código.

3 La herramienta MoteGen

La herramienta *MoteGen* ha sido desarrollada haciendo uso del entorno de desarrollo Eclipse [7], el *framework* de modelado EMF (*Eclipse Modeling Framework*)[8] y otras tecnologías que se soportan en ambos como GMF (*Graphical Modeling Framework*), AGG (*The Attributed Graph Grammar System*) [9] y MOFScript[10].

En la figura 1 se puede apreciar el aspecto de la herramienta. En ella además se observa cómo el DSL que se ha desarrollado [11] describe una aplicación. La aplicación en cuestión se basa, por un lado, en la especificación de grupos de nodos y las comunicaciones entre ellos mediante enlaces inalámbricos y, por otro lado, en la especificación del comportamiento de estos nodos (todos los nodos pertenecientes a un mismo grupo se comportarán igual). El comportamiento se expresa mediante la interconexión de unidades funcionales, encargadas de las tareas básicas que se ejecutan en los nodos, las cuales acceden a los recursos de éstos (memoria – a través de variables simples así como de tipos de datos complejos, sensores, parámetros de configuración y entradas y salidas de los microcontroladores de cada uno de los nodos de la red). La aplicación modelada en la figura a modo de ejemplo toma mediciones de luz en dos nodos de la red (NG1 y NG2) y cuando el valor tomado en NG2 sea el mayor se encenderá un led en este nodo, en caso contrario se apagará. Esta aplicación es muy simple aunque también se han modelado aplicaciones más complejas con la herramienta. Como ejemplo se pueden citar diversas aplicaciones de monitorización de datos desde un PC y aplicaciones de localización de personas u objetos.

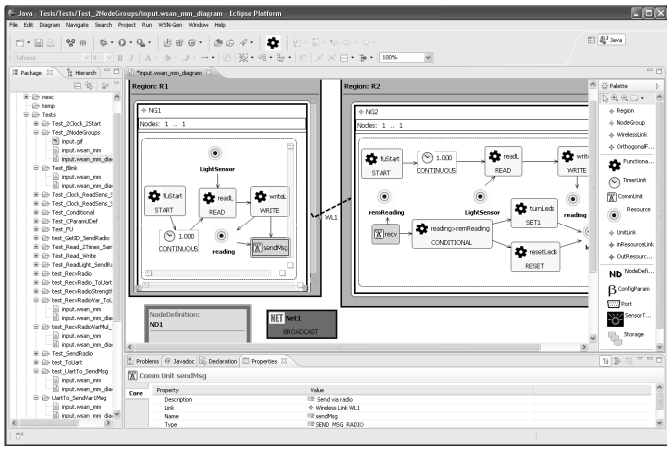


Fig. 1. Captura de la herramienta *MoteGen*.

Para la generación de código se ha recurrido a un esquema con tres niveles de modelos de acuerdo al estándar MDA. En un nivel más alto se encuentran los modelos PIM correspondientes al LED para redes de sensores. En un segundo nivel de menor abstracción se sitúan modelos descritos con el lenguaje de modelado V3-CM [12], un lenguaje de modelado genérico orientado a componentes que se basa en tres vistas: la primera de ellas describe la estructura de componentes del sistema y la segunda y tercera el comportamiento de los componentes mediante máquinas de estados y diagramas de actividades. Finalmente se cuenta con un último nivel con modelos PSM correspondientes a la plataforma de ejecución TinyOS-nesC. Las transformaciones entre los distintos niveles se realizan mediante el lenguaje de transformación AGG. Para generar código se parte de los modelos PSM y, mediante una transformación con MOFScript, se obtiene código nesC que hace uso de componentes preexistentes del sistema operativo TinyOS.

3 Conclusiones y trabajos futuros

Se ha presentado la herramienta MoteGen usada para desarrollar aplicaciones para redes de sensores inalámbrica. Esta herramienta sigue un proceso de Desarrollo de Software Dirigido por Modelos en el que las aplicaciones son descritas mediante un Lenguaje Específico de Dominio para redes de sensores. Mediante sucesivas transformaciones de estas descripciones se obtiene código ejecutable en el lenguaje nesC del sistema operativo TinyOS. Las ventajas que introduce el uso de la herramienta son considerables tanto en reducción de tiempo de aprendizaje como en el tiempo de desarrollo de aplicaciones. Sin embargo aún no se ha cuantificado claramente esta mejora ya que es muy variable en función de los conocimientos del desarrollador y de las características de la aplicación que se modela.

El estado actual de desarrollo de la herramienta está bastante avanzado, habiéndose logrado el objetivo principal de la generación de código. El código generado no se

limita a interconectar componentes existentes, sino que además se generan nuevos componentes de control encargados de coordinar al resto de componentes y realizar diversos tipos de procesado.

Aún quedan numerosos aspectos que mejorar en la herramienta que van desde la ampliación de la funcionalidad del nuevo lenguaje (principalmente las comunicaciones) hasta la mejora de su usabilidad. A este respecto se pretende mejorar el interfaz de usuario de la herramienta así como introducir restricciones en el lenguaje que aseguren que no existan errores sintácticos en sus descripciones. También se está estudiando la posibilidad de ampliar la herramienta con otros servicios como la simulación de aplicaciones directamente sobre las descripciones del LED o la integración de éstas con otras tecnologías distintas de las redes de sensores inalámbricas (domótica y robótica).

Agradecimientos. Los autores desean expresar su agradecimiento a la Fundación Séneca de la Región de Murcia Region (ref ID-02998/PI/05) y al Ministerio de Educación y Ciencia, CICYT MEDWSA (ref TIN1006-15175-C05-02), por haber financiando parcialmente este trabajo. Esta investigación también ha sido parcialmente financiada por el proyecto TEC2007-67966-01/TCM (CON-PARTE-1).

Referencias

1. Deursen, A.v., Klint, P., Visser, J.: Domain-Specific Languages: An Annotated Bibliography, SIGPLAN Notices 35(6), 26--36 (2000)
2. Kent, S., Model Driven Engineering. In: IFM 2002, LNCS, vol. 2335, pp.286--298, Springer, Heidelberg (2006)
3. MDA guide v1.0.1, http://www.omg.org/docs/omg/03-06-01.pdf
4. Akyildiz, I. A., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A Survey of Sensor Networks. IEEE Communications Magazine 40, 8, pp. 102--114. IEEE Press (2002)
5. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister K.: System Architecture Directions for Networked Sensors. In: International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 93--104, ACM Press, Cambridge (2000)
6. Gay, D. Levis, P. Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesC language: A holistic approach to networked embedded systems, In: Programming Language Design and Implementation, pp. 1--11, ACM Press, California (2003)
7. Eclipse, http://www.eclipse.org
8. EMF (Eclipse Modeling Framework), http://www.eclipse.org/emf/
9. AGG: The Attributed Graph Grammar System, http://tfs.cs.tu-berlin.de/agg
10. Oldevik, J., Neple, T. , Gronmo, R., Aagedal, J., Berre, A.: Toward Standardised Model to Text Transformations, In: ECMDA-FA 2006, LNCS, vol. 3748, pp. 239--253. Springer, Heidelberg (2005)
11. Losilla, F., Vicente-Chicote, C., Alvarez, B., Iborra, I., Sánchez, P.: Wireless sensor network application development: An architecture-centric MDE approach. In: ECSA 2007, LNCS, vol. 4758, pp. 179--194 (2007)
12. Alonso, D., Vicente-Chicote, C., Barais, O.: V3Studio: A Component-Based Architecture Modeling Language. In: 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pp. 346--355, IEEE Press (2008)

Generación de herramientas de modelado colaborativo independientes del dominio

Jesús Gallardo, Crescencio Bravo, Miguel Ángel Redondo

Escuela Superior de Informática
Departamento de Tecnologías y Sistemas de Información
Universidad de Castilla-La Mancha
{jesus.gallardo; crescencio.bravo; miguel.redondo}@uclm.es

Abstract. Dentro del área de las herramientas groupware, un campo de gran utilidad y con numerosas aplicaciones tanto en la industria como en la enseñanza son las herramientas colaborativas de modelado. Sin embargo, estas herramientas normalmente son difíciles de construir y se dedican a un dominio de modelado concreto. Como respuesta a esta problemática, hemos desarrollado un método de desarrollo dirigido por modelos para la generación de este tipo de herramientas. Este método puede ser usado por cualquier usuario independientemente de su conocimiento de la programación de aplicaciones groupware. El método utiliza como marco tecnológico la funcionalidad de los plug-ins de Eclipse que conforman el *Eclipse Modeling Project*. En concreto, se ha extendido la funcionalidad del *Graphical Modeling Framework* (GMF) para que las herramientas generadas sean herramientas colaborativas y para hacer más sencilla la definición de los dominios de diseño.

Keywords: Desarrollo de Software Dirigido por Modelos, Groupware, CSCW.

1 Introducción y motivación

La sociedad actual cada vez tiene más necesidades de comunicación y de trabajo colaborativo en las diferentes tareas que se llevan a cabo en el lugar de trabajo, y también en las actividades de aprendizaje e incluso en los momentos de ocio. Esta situación requiere de la aparición de un nuevo tipo de software diseñado para el trabajo en grupo, que es habitualmente conocido como *groupware*. En el presente trabajo nos hemos centrado en un tipo concreto de aplicaciones *groupware*, como son las herramientas colaborativas de modelado síncrono. En dichas herramientas, varios diseñadores interactúan de manera distribuida para la construcción de un modelo o artefacto. Existen numerosas aplicaciones *groupware* que permiten desarrollar modelos o artefactos relativos a un dominio de aplicación concreto. Por ejemplo, DomoSim-TPC [1], que trabaja sobre el dominio de la Domótica, o Co-Lab [2], que lo hace sobre la Dinámica de Sistemas.

En determinadas circunstancias, como en procesos de construcción o ingeniería más o menos complejos, se presenta la necesidad de trabajar durante el ciclo de vida de un proyecto con numerosos tipos de diagramas diferentes. Si se desea desarrollar

esos modelos o diagramas de forma colaborativa, la cantidad de herramientas que se debe manejar es muy grande, y cada una con sus particularidades de uso y su propia curva de aprendizaje. Por lo tanto, se presenta como una idea interesante el disponer de herramientas colaborativas de diseño independientes del dominio, en las cuales el modelo o artefacto a crear no esté restringido a un dominio de diseño concreto. En lugar de ello, la herramienta será capaz de trabajar sobre varios ámbitos de diseño descritos por medio de algún procedimiento de configuración previo. Ya existen herramientas colaborativas de modelado independientes del dominio, como CoolModes [3], Synergo [4] o SPACE-DESIGN [5], aunque algunas de ellas no son realmente independientes del dominio, sino que están programadas para trabajar con un conjunto cerrado de dominios de aplicación. Otra carencia importante de la mayoría de herramientas es su carencia de elementos de soporte al *awareness* (conocimiento y percepción de los compañeros y de su trabajo) y a la comunicación, tales como telepunteros o chats. También existen herramientas de modelado independientes del dominio, pero que no fueron concebidas para soportar colaboración síncrona, como AToM3 [6].

A la hora de implementar un método de desarrollo que dé soporte a la creación de este tipo de herramientas, el uso de técnicas de metamodelado [7] va a ser sumamente útil. Mediante el metamodelado, se definirá la estructura de los modelos a construir con la herramienta, así como la estructura de la propia herramienta. De este modo, existirá un conjunto de meta-modelos que será la base para construir los modelos que caractericen una herramienta concreta [8]. Por tanto, nuestro método va a evitar los problemas que supondría rediseñar la herramienta para cada nuevo dominio de aplicación con el que se deseara trabajar, pudiendo entonces emplear la misma herramienta para manejar varios dominios diferentes entre sí. Además, el método de desarrollo va a ser poder usado por usuarios sin conocimiento específico acerca de la programación de aplicaciones software gracias a la simplicidad del proceso de definición de nuevos dominios de aplicación y de configuración de las herramientas a generar.

2 Un método de desarrollo de herramientas colaborativas de modelado independientes del dominio

El método de desarrollo de aplicaciones colaborativas síncronas de modelado aquí propuesto es un método dirigido por modelos, y basado en el enfoque de metamodelado integrado en la plataforma Eclipse en el *Eclipse Modeling Project*. Este proyecto está formado por varios *plugins,* entre los que destacan el *Eclipse Modeling Framework* (EMF) y el *Graphical Modeling Framework* (GMF). Se ha elegido dicho enfoque, principalmente, por dos motivos: por un lado, por el soporte que proporciona el *plugin* GMF en cuanto a la automatización del desarrollo de la herramienta final, sobre todo en lo que se refiere a la implementación de la Interfaz Gráfica de Usuario; y por el otro lado, por las posibilidades que otorga Eclipse a la hora de servir como plataforma contenedora de las distintas herramientas que puedan formar parte del marco tecnológico del método de desarrollo propuesto. Eclipse permite soportar todo el ciclo de desarrollo del software y, gracias a su grado de

implantación en la industria, se ha convertido en una plataforma ampliamente usada para soportar los procesos asociados al desarrollo de software.

Las herramientas que genera el *plugin* GMF son herramientas monousuario, y la definición de los dominios al usar GMF de la forma habitual no es trivial para un usuario no avanzado, ya que implica la definición de diversos modelos correspondientes a otros tantos metamodelos definidos por GMF. De este modo, nuestra contribución es, por un lado, integrar la funcionalidad colaborativa en las herramientas que genera GMF, y, por el otro, facilitar la definición de los dominios de aplicación, trabajando con un único modelo de dominio, además de otro modelo para definir el espacio de trabajo. En la figura 1 se muestra un ejemplo de herramienta que podría ser generada a partir del método de desarrollo propuesto.

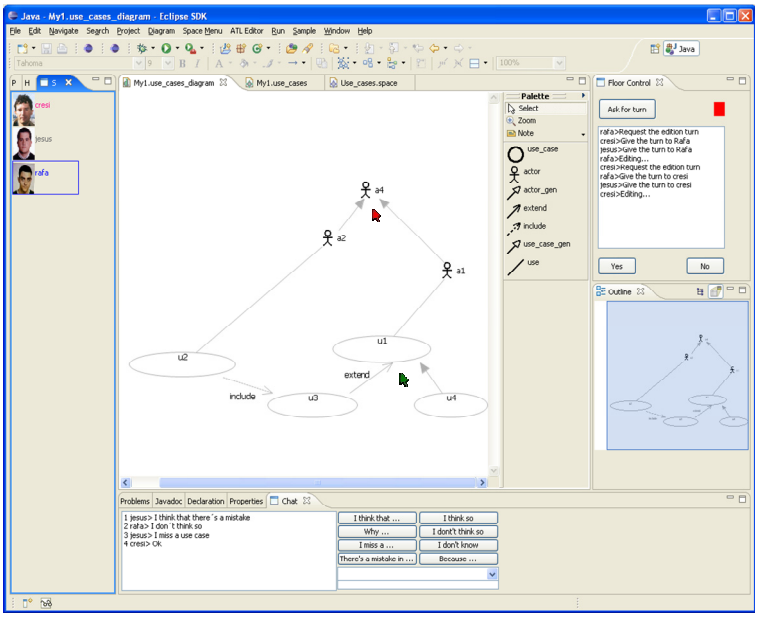


Fig. 1. Herramienta generada a partir del método de desarrollo propuesto.

El método propuesto se basa en tres marcos: un marco metodológico (Figura 2), un marco conceptual y un marco tecnológico. Mediante estos tres marcos se proporciona soporte completo al método de desarrollo. El marco metodológico consiste en una serie de fases que deben ser seguidas por el usuario no experto que desee desarrollar una herramienta colaborativa de modelado a partir de nuestro método de desarrollo. Estas fases son: (i) la identificación del dominio; (ii) el modelado del dominio y del espacio de trabajo, incluyendo los elementos de *awareness*, coordinación y comunicación que se desee incluir en el sistema final; (iii) la producción de la herramienta *groupware* final, que abarca las transformaciones entre modelos y la generación propiamente dicha de la herramienta; y (iv) el uso de la herramienta generada. Por su parte, el marco conceptual está formado por los modelos empleados en el proceso de metamodelado. Y finalmente, el marco tecnológico consiste en una serie de *plug-ins* para la plataforma Eclipse que han sido modificados para integrar la

funcionalidad colaborativa deseada y han sido integrados para dar soporte al método de desarrollo en su totalidad.

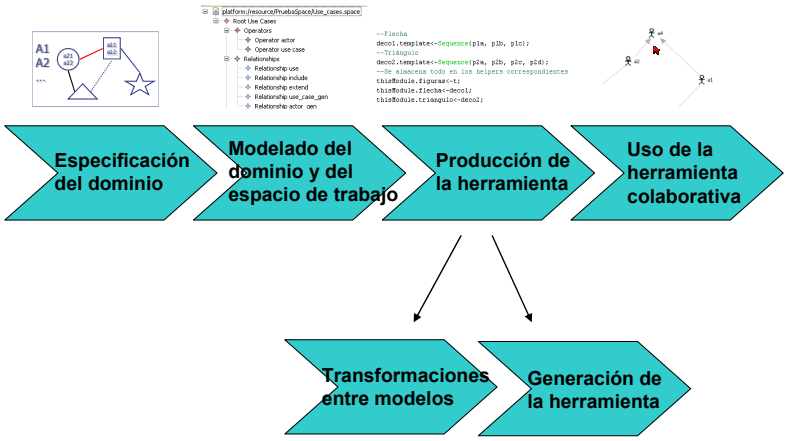


Fig. 2. Marco metodológico del método de desarrollo.

Referencias

1. Bravo, C., Redondo, M.A., Ortega, M., Verdejo, M.F.: Collaborative environments for the learning of design: A model and a case study in Domotics. Computers and Education, 46 (2), pp. 152-173 (2006)
2. van Joolingen, W.R., de Jong, T., Lazonder, A.W., Savelsbergh, E.R., Manlove, S.: Co-Lab: research and development of an online learning environment for collaborative scientific discovery learning. Computers in Human Behavior, 21, pp. 671-688 (2005)
3. Pinkwart, N., Hoppe, U., Bollen, L., Fuhlrott, E.: Group-Oriented Modelling Tools with Heterogeneous Semantics. In: Cerri, S., Gouardères, G., Paraguaçu, F. (eds.) Intelligent Tutoring Systems, LNCS, vol. 2363. Springer, Heidelberg (2002)
4. Avouris, N., Margaritis, M., Komis, V.: Modelling interaction during small-groups synchronous problem-solving activities: The Synergo approach. In: Proceedings of the 2nd International Workshop on Designing Computational Models of Collaborative Learning Interaction (2004)
5. Duque, R., Gallardo, J., Bravo, C., Mendes, A.J.: Defining tasks, domains and conversational acts in CSCW systems: the SPACE-DESIGN case study. Journal of Universal Computer Science, vol. 14, no. 9, pp. 1463-1479 (2008)
6. de Lara, J., Vangheluwe, H.: Defining Visual Notations and Their Manipulation Through Meta-Modelling and Graph Transformation. Journal of Visual Languages and Computing. Special section on "Domain-Specific Modeling with Visual Languages", Vol 15(3-4), pp. 309-330 (2004)
7. Favre, J.M.: Towards a Basic Theory to Model Model Driven Engineering. In: Proceedings of Workshop on Software Model Engineering, WISME 2004 (2004)
8. Gallardo, J., Bravo, C., Redondo, M.A.: An ontological approach for developing domain-independent groupware. In: Proceedings of WETICE 2007, pp. 206-207. IEEE Computer Society (2007)

ModelSET Component Framework: Refinando el Ciclo de Vida de MDA

E. Victor Sánchez Rebull, Orlando Avila-García, Pablo J. Hernández López, Salvador Martínez Pérez, Antonio Estévez García

Open Canarias, S.L.
C/. Elías Ramos González, 4 - Oficina 304
38001 Santa Cruz de Tenerife, Spain
{vsanchez,orlando,pablojhl,smartinez,aestevez}@opencanarias.es
http://www.opencanarias.es

1. Introducción

La aproximación Model-Driven Architecture propone obtener modelos PSM (*Platform-Specific Model*) a partir de modelos PIM (*Platform-Independent Model*), mediante la ejecución de transformaciones de modelos que automatizan el proceso. Un PIM es una especificación formal de la estructura y comportamiento de un sistema del que se omiten detalles técnicos dependientes de la plataforma de destino. Un PSM representa el mismo sistema a un nivel de abstracción inferior, dado en términos de una plataforma tecnológica concreta. Así, se facilita la generación de código fuente u otros artefactos derivados a partir del PSM.

Uno de los beneficios más destacables de este ciclo de vida MDA es la portabilidad de las aplicaciones mediante la realización de un mismo modelo sobre distintas plataformas. Ello permite la integración de diferentes aplicaciones mediante la interrelación explícita entre sus respectivos modelos y promueve el soporte a la evolución de los sistemas.

La guía de MDA [1] menciona el concepto *Platform Model*. El término *plataforma* en MDA hace referencia a los detalles tecnológicos y de ingeniería que pueden ser considerados irrelevantes de cara a la definición de los aspectos funcionales de los componentes software. En cambio, estos detalles se centran en proporcionar la información que determina cómo un componente debe ser implementado para su óptimo funcionamiento en entornos tecnológicos específicos.

La aproximación MDA típica selecciona un metamodelo PIM y uno PSM y establece relaciones entre ambos para automatizar la conversión de modelos PIM a PSM por medio de transformaciones de modelos. Si se introduce el concepto de modelos de plataforma encontramos el clásico flujo en forma de "Y" con modelos de componentes software (PIM) en la punta superior izquierda y modelos de plataformas en la punta superior derecha. Una transformación de modelos los mezcla para producir los modelos PSM resultantes.

La transformación se convierte entonces en una especie de *mapping* o *weaving* entre conceptos del modelo del sistema y conceptos de plataforma, como sucede en [2]. La transformación es dependiente del metamodelo PIM y del metamodelo con el cual se definen los modelos de plataformas. Esto obliga a codificar el metamodelo del PSM

Actas JISBD 2009, pp.174-177, ISBN: 978-84-692-4211-7

a generar en el modelo de plataforma, no puede ser referenciado directamente por la transformación en cuestión, para preservar su independencia y reutilización para todos los metamodelos de PSM posibles.

El prototipo *ModelSET Component Framework* (MCF) descrito en esta propuesta gira en torno al modelado de sistemas estructurados en componentes, descritos usando el lenguaje de modelado UML. Este prototipo propone una aproximación novedosa a la definición de plataformas y su aplicación al ciclo MDA. Se utilizan modelos de plataforma, pero se prescinde del uso de modelos PSM. Los modelos de plataforma contienen la información requerida para obtener artefactos de bajo nivel, tales como código fuente, naturalmente compatible con la plataforma utilizada, y por lo tanto, ejecutable en ella. Estos artefactos se obtienen en MCF mediante la aplicación de un proceso fundamentalmente automático, que puede requerir de cierta interacción con el usuario de cara a la toma de decisiones.

2. Modelado del PIM

El modelo PIM que representa el sistema software se define mediante UML. Dado que este lenguaje es muy amplio y no provee por sí mismo una metodología para la construcción e integración de los diferentes aspectos de un sistema, se ha diseñado una metodología propia basada en las vistas 4+1 de Kruchten. Usando como plataforma base Eclipse, y para la edición de los modelos UML los componentes UML2 y UML2Tools, se define una perspectiva Eclipse por cada una de las vistas de Kruchten. Cada perspectiva muestra la información y los editores de modelos UML pertinentes para el punto de vista concreto.

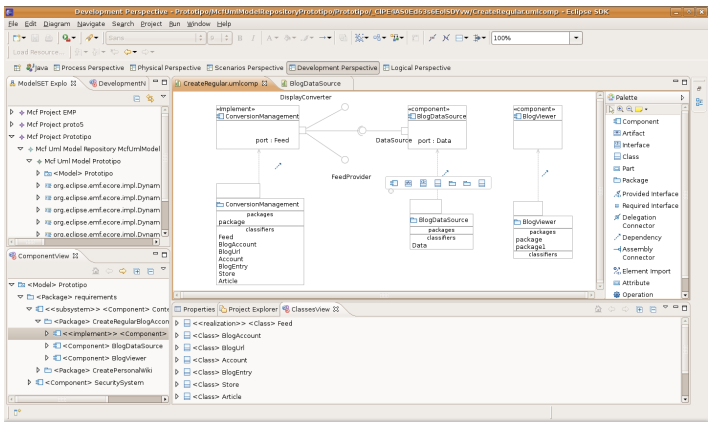


Figura 1. Entorno típico de edición de modelos de componentes UML (PIM)

La Fig. 1 muestra un entorno de edición de modelos UML en la herramienta MCF. Nuestra metodología propone el siguiente patrón: Los requisitos del sistema son modelados como casos de uso, que se implementan mediante componentes contenidos en

paquetes. Cada componente se realiza mediante un conjunto de paquetes de implementación que contendrán sus respectivas clases e interfaces. Finalmente, el código fuente de estos artefactos podrá ser generado en un proceso que requiere la intervención de modelos de plataforma.

Es interesante resaltar que, siguiendo esta metodología, podremos mantener la trazabilidad entre los requisitos y los componentes y entre éstos y los paquetes de implementación. La decisión de utilizar UML como único lenguaje de modelado PIM aporta varios beneficios, como el aumento de la capacidad de integración de los modelos en la herramienta MCF y el fomento de la reutilización de todas las partes que la componen.

3. Modelado de Plataformas

Los modelos de plataforma describen la plataforma a través de un lenguaje diseñado para tal fin. Así, en lugar de ocuparse en metamodelar dominios PSM a partir de MOF, que ofrece conceptos tan abstractos como *Class*, *Attribute*, *Package*, etc., modelamos plataformas directamente, en un lenguaje específico de dominio, que proporciona terminología adaptada a los elementos a modelar.

El lenguaje de definición de plataformas se ha subdividido en varios sublenguajes. El primero, llamado *Platform Definition*, representa definiciones de plataforma organizadas por tipos ontológicos (ver [3]). Estos tipos pueden corresponder a conceptos hardware o software indistintamente. Los tipos más relevantes con que se está experimentando incluyen: *Sistema Operativo* (e.g., Linux, sus distribuciones, versiones específicas), *Máquina Virtual* o *Librería*.

El segundo sublenguaje, *Platform Usage*, representa familias de software instalables en diversas máquinas computacionales o nodos. Los elementos de un modelo *Usage* determinan posibles usos y combinaciones de los elementos definidos en un modelo *Definition*. Las familias definidas deben ser configuradas antes de obtener una plataforma usable para la generación de artefactos a partir del modelo PIM. Para ello se usa un tercer sublenguaje, *Platform Configuration*.

La plataforma así obtenida representa un conjunto de configuraciones de software a instalar en distintos entornos de ejecución o nodos de un sistema distribuido. Es en estos entornos donde los artefactos software generados a partir del sistema modelado podrán ser instalados y ejecutados.

4. Generación

El proceso de generación de estos artefactos, toma como información de entrada el modelo UML y un modelo de plataforma donde se especifican las relaciones entre los conceptos del modelo y las acciones a realizar. Estas acciones están descritas como plantillas MTL [4], implementación del lenguaje estándar de transformaciones de modelo a texto MOF2Text de OMG [5].

Durante el proceso de generación automática, se seleccionan las acciones definidas para el modelo de plataforma y se aplican a los elementos del modelo UML, solicitando interacción del usuario cuando es necesario. Esto resulta útil para tratar situaciones

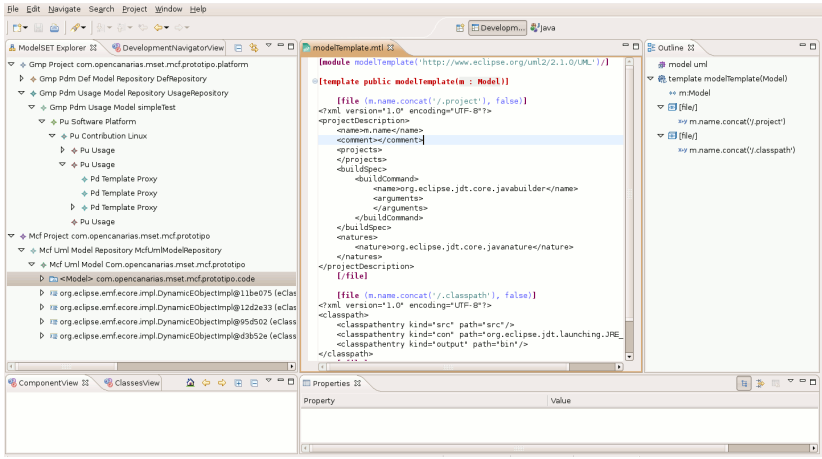


Figura 2. Entorno de edición de modelos de plataforma y de las plantillas de generación

de variabilidad que no se pueden resolver con la información contenida en el modelo UML. En la Fig. 2 se observa el entorno de trabajo, donde se define un modelo de plataforma para una aplicación sobre Linux, que contendrá el conjunto de plantillas a aplicar durante la generación de la aplicación final. El entorno está diseñado para trabajar de un modo sencillo con el modelo UML, así como con las plantillas MTL.

La generación sigue un proceso basado en ordenación compositiva de plantillas. En aproximaciones MDA más tradicionales, las transformaciones PIM-PSM son de carácter monolítico, han de reescribirse casi por completo para cada dominio destino. En su lugar, el carácter compositivo y granular de las plantillas usadas en el entorno MCF favorecen la actualización incremental de las definiciones de plataforma junto a la capacidad para incorporar nuevos conceptos de plataforma en representación de tecnologías emergentes.

Referencias

1. OMG: MDA guide version 1.0.1. Final Adopted Specification omg/2003-06-01, OMG (Jun 2003) http://www.omg.org/docs/omg/03-06-01.pdf.
2. Belangour, A., Bézivin, J., Fredj, M.: The Design Decision Assistant Tool: A Contribution to the MDA Approach. In: Proceedings of the 1ere Journée sur l'Ingénierie Dirigée par les Modèles. (Oct 2005)
3. Atkinson, C., Kühne, T.: Multi-Level Platform Descriptions. In: Proceedings of the 1st Workshop on Metamodeling and Corresponding Tools. (March 2005)
4. Eclipse: MTL Project http://www.eclipse.org/modeling/m2t/.
5. OMG: MOF Model to Text Transformation Language. Technical Report formal/2008-01-16, OMG (Jan 2008) http://www.omg.org/docs/formal/08-01-16.pdf.

MOMENT2
EMF Model Transformations in Maude

Artur Boronat[1] and José Meseguer[2]

[1] Department of Computer Science, University of Leicester
[2] Department of Computer Science, University of Illinois at Urbana-Champaign

MOMENT2 [1] is a tool suite that provides support for formal model-driven development using the Eclipse Modeling Framework (EMF) [2] as user front-end, and the Maude framework [3] as its formal back-end. The mathematical foundations of this tool suite directly rely on an executable, reflective, algebraic semantics of MOF metamodels[4,5], which enables the automatic formalization of EMF metamodels as membership theories and its models as graphs, and a rewriting logic semantics for MOF model transformations [6].

MOMENT2 is an executable, reflective, algebraic MOF framework where a metamodel $\mathscr{M}$ is automatically formalized as a membership equational theory $\mathbb{A}(\mathscr{M})$ providing the notions of: *model type* $\mathscr{M}$, as a set of well-formed models; *metamodel realization* as the theory $\mathbb{A}(\mathscr{M})$ that constitutes the metamodel specification; and *metamodel conformance* $M : \mathscr{M}$, as the membership of a term M that represents a model to the model type $\mathscr{M}$. MOMENT2 also provides an algebraic specification of the OCL language that permits the definition of richer metamodels with OCL constraints, called *metamodel specifications* in our approach [5]. A metamodel specification $(\mathscr{M}, \mathscr{C})$ is a pair of a metamodel $\mathscr{M}$ and a set $\mathscr{C}$ of OCL constraints that is formalized as the theory $\mathbb{A}(\mathscr{M})$ enriched with equationally-defined functions, which present the OCL constraints $\mathscr{C}$ as boolean predicates. These predicates are used as conditions in a membership that defines constrained model type $(\mathscr{M}, \mathscr{C})$ so that a model M is semantically valid, $M : (\mathscr{M}, \mathscr{C})$, if it is well-formed, i.e., $M : \mathscr{M}$, and satisfies the OCL constraints $\mathscr{C}$.

On the other hand, MOMENT2 provides a tool for defining, formalizing, executing and analysing EMF *model transformations* using Maude and its execution and formal verification techniques. The MOMENT2 model transformation language is syntactically based on the QVT Relations language, for defining model patterns, and semantically based on the Single-Pushout Approach, borrowed from graph transformation theory [7]. A pair $(\mathscr{M}, \mathscr{T})$, of a MOF metamodel $\mathscr{M}$ and a MOMENT2 model transformation definition $\mathscr{T}$, represents a model transformation, whose *semantics* is formally defined by a rewrite theory $\mathbb{R}(\mathscr{M}, \mathscr{T})$ such that $\mathbb{A}(\mathscr{M}) \subseteq \mathbb{R}(\mathscr{M}, \mathscr{T})$. Applications of MOMENT2 include:

- Automatic formalization of domain-specific modeling languages, whose abstract syntax is defined as a metamodel and whose behavioral specification is given as an in-place model transformation. In this case, the model transformation becomes an interpreter for the language.
- Formalization of model-based systems, where the structure of the system is given as a metamodel and its dynamics is given as an in-place model transformation. In

Actas JISBD 2009, pp.178-179, ISBN: 978-84-692-4211-7

this case, the model transformation corresponds to the behavioral specification of the system.
- Analysis of in-place model transformations. MOMENT2 leverages the use of Maude's reachability analysis and LTL model checker to work with model patterns and model transformations.
- Definition of multi-model transformations that may involve several models, which may or may not conform to the same metamodel. Such model transformations may take several input models and may produce several output models.
- Static analysis of models by using OCL constraints.
- The MOMENT2 tool suite is built on top of the EMF and it can be used for formalizing modeling environments, either domain-specific, such as OSATE-AADL [8], or generic, such as UML [9].
- The front-end for defining model transformations is developed with openArchitectureWare and it is therefore based on a metamodel. That is, model transformation definitions are also MOMENT2 models. This enhances the definition of higher-order multi-model transformations.

References

1. MOMENT2 web site: (2009) `http://www.cs.le.ac.uk/~aboronat/tools/moment2`.
2. Eclipse Organization: The Eclipse Modeling Framework (2007) `http://www.eclipse.org/emf/`.
3. Clavel, M., Durán, F., Eker, S., Meseguer, J., Lincoln, P., Martí-Oliet, N., Talcott, C.: All About Maude. Springer LNCS Vol. 4350 (2007)
4. Boronat, A., Meseguer, J.: An Algebraic Semantics for MOF. In: FASE. Volume 4961 of LNCS., Springer (2008)
5. Algebraic Semantics of OCL-Constrained Metamodel Specifications. In: 47th International Conference, Objects, Models, Components, Patterns. TOOLS-EUROPE 2009. Zurich, 29 June - 3 July 2009. Volume 33 of LNBIP., Springer (2009) To appear.
6. Boronat, A., Heckel, R., Meseguer, J.: Rewriting Logic Semantics and Verification of Model Transformations. In: FASE. Volume 5503 of LNCS., Springer (2009)
7. Rozenberg, G.: Handbook of Grammars and Computing by Graph Transformation, Vol. 1. World Scientific Publishing Company (1997)
8. SAE: AADL (2007) `http://www.aadl.info/`.
9. Eclipse Organization: Model development tools (2007) `http://www.eclipse.org/modeling/mdt/`.

Parte VI

Sesión 5. BBDDs y Tecnologías de SGBD

Philo: un Sistema Experimental de Gestión de Bases de Datos Distribuido en Memoria de Alto Rendimiento

Alejandro Bascuñana Muñoz[1], Jesús Javier Arauz[1,]

[1] Unidad de Tecnología e Innovación, Centro de I+D Ericsson España
Vía de los Poblados, 13, 28033 Madrid (España)
{Alejandro.Bascunana, Jesus.Javier.Arauz}@ericsson.com

Abstract. Actualmente la mayoría de los nodos que almacenan información de usuarios en las redes de telecomunicaciones móviles y convergentes (HLR, AuC, HSSz, etc.) lo hacen utilizando bases de datos propietarias empotradas (con la lógica fuertemente acoplada a los datos), no existiendo sincronización alguna entre dichas bases de datos. Como consecuencia, existen múltiples copias de los mismos datos, lo que origina inconsistencias e induce costes añadidos en la gestión de los mismos. Por todo ello, la consolidación de datos se ha convertido en un asunto de la máxima prioridad para los operadores de telecomunicación y suministradores de equipamiento, ya que garantiza una disminución de los costes operativos (OPEX) y de adquisición (CAPEX) del operador, además de una simplificación de la operativa asociada a la gestión de los datos de usuario. Para poder llegar a implementar este paradigma, los nodos anteriormente mencionados deben permitir el almacenamiento de la información en bases de datos externas al propio nodo (esto es, desacoplando lógica y datos) de forma que se permita la compartición de datos entre dichos nodos, permitiendo el acceso a los datos de usuario a terceras partes mediante protocolos estándar. Los requisitos de rendimiento, alta disponibilidad, escalabilidad, etc. que deben cumplir estos repositorios comunes hacen que no se pueda recurrir a bases de datos de propósito general y que sea necesario utilizar productos con características concretas. En este artículo se hace un resumen del estado del arte existente, las funcionalidades y los requisitos necesarios para implementar una base de datos de estas características y los criterios arquitectónicos elegidos para la implementación de un prototipo inicial de base de datos en memoria. El objetivo del proyecto es proporcionar un prototipo que desarrolle las funcionalidades mínimas con el que validar los conceptos previamente expuestos y para poder utilizarlo como plataforma para posteriores proyectos de I+D. Se incluyen también datos preliminares de rendimiento obtenidos en las diferentes pruebas de sistema realizadas.

Keywords. Bases de Datos Distribuidas, Alta Disponibilidad, Escalabilidad Lineal, Consolidación de Datos, OPEX, CAPEX, LDAP, SQL, JDBC.

Actas JISBD 2009, pp.183-194, ISBN: 978-84-692-4211-7

1 Introducción

Las redes de telecomunicaciones, y en concreto las redes móviles e IMS (IP Multimedia Subsystem), disponen de un conjunto de nodos que mantienen la información actualizada de los subscriptores y que permiten gestionar la autorización de accesos, cifrado de comunicaciones, y en definitiva la realización de múltiples operaciones que tienen en cuenta la información específica para cada uno de los usuarios registrados en la misma. Estos nodos son el HLR (Home Location Register), el HSS (Home Subscriber Server), el AuC (Authentication Center), y en definitiva todos aquellos dedicados a gestionar este tipo de información. Estos son los pilares fundamentales sobre los que se basan la seguridad, el control de acceso, la facturación, el cambio de localización de los usuarios, etc.

Para almacenar esta información tradicionalmente se han utilizado bases de datos propietarias empotradas dentro de los propios nodos, hecho que ha impedido la compartición de la información y por tanto la administración única de los mismos. La existencia de múltiples copias de un mismo dato ha generado unos altos costes de operaciones (OPEX) y de adquisición (CAPEX) que necesitan ser disminuidos en el futuro. Por lo anteriormente expuesto, se ve la necesidad de tener una solución que pueda proporcionar los medios para realizar la consolidación de los datos existentes en los diferentes nodos (HLR, HSS, AuC, etc.) en una única base de datos común, para que de esta manera los costes asociados decrezcan y se sigan manteniendo altos los estándares de fiabilidad, tiempo de respuesta y otros requerimientos críticos. En este entorno la utilización de bases de datos de tiempo real que puedan hacer frente a estos requisitos impuestos por las aplicaciones es algo que se torna en fundamental.

El principal reto al que se enfrenta una solución de base de datos en este tipo de entornos es el de los requisitos de tiempo real a los que se tiene que hacer frente. Como ejemplo se manejan transacciones con tiempos de acceso medios máximos menores a 10ms, volúmenes de usuarios variables siempre del orden de millones de abonados. Se requiere escalabilidad lineal, alta redundancia 99,999% (“five nines availability”), redundancia geográfica de los datos, además de múltiples requisitos específicos de cada una de las aplicaciones.

Como se verá posteriormente la solución viene de la mano de sistemas distribuidos en memoria combinados con desarrollos en entornos abiertos y utilizando máquinas con procesadores x86.

En el presente artículo se hace un breve recorrido por el estado del arte existente en el campo y posteriormente se introducirán las decisiones arquitectónicas tomadas en esta primera fase del proyecto. A continuación se hace una descripción de las herramientas de desarrollo que están siendo utilizadas y de las máquinas desplegadas en el laboratorio. Finalmente se proporciona un resumen de los primeros resultados obtenidos y una evaluación de los mismos contrastándolos con los requisitos impuestos. Los resultados presentados son preliminares y se irán complementando con futuras investigaciones e implementaciones.

2 Estado del Arte

Los requisitos en cuanto a alta disponibilidad, tiempo de respuesta, escalabilidad, etc. son dificilmente alcanzados en su totalidad por las bases de datos de propósito general que gozan de mayor penetración en el mercado, como Oracle 11g [1], MySQL [2], etc. Por ello surgen sistemas específicos de bases de datos distribuidas en memoria de proveedores como Xeround [3], Apertio [4] y MySQL Cluster [5]. Como puede observarse de las referencias anteriores, las bases de datos que cumplen estos requisitos son bases de datos en memoria, que permiten alcanzar unos tiempos de acceso menores a 10ms, y distribuidas, para asegurar la escalabilidad y alta disponibilidad de los datos.

La mayoría de los sistemas de BBDD en memoria existentes cumplen la gran parte pero no todos los requisitos mencionados en la sección anterior. El producto que más se aproxima a dichos requisitos es el indicado en [3].

Una primera propuesta de solución fue expuesta por Stonebraker (MIT) and Helland (Microsoft) en "The End of an Architectural Era (It's Time for a Complete Rewrite)" [6]. La base fundamental sobre la que, según los autores, debe basarse el nuevo paradigma de las bases de datos es la siguiente:

- Todo el código y las estructuras de las bases de datos actuales deben ser eliminados porque las aplicaciones actuales no tienen nada que ver con las existentes hace 40 años.
- Se propone una nueva arquitectura dos órdenes de magnitud más rápida que los sistemas tradicionales basados en accesos a disco.
- Todos los datos deben de residir en memoria (MMDB).
- Los procesos deben ser monohebra para evitar la introducción de más complejidad debido a los cambios de contexto.
- La sobrecarga impuesta por el manejo de REDO logs y de check points debe ser eliminada.
- Los nodos no deben compartir en ningún momento ningún tipo de datos, arquitecturas "Shared Nothing"
- Las bases de datos no deben de requerir una puesta a punto por parte de expertos. Debe ser posible utilizarlas sin un mantenimiento realizado por personal experto.
- SQL no es necesariamente la respuesta.

Los requisitos que han guiado el diseño de la arquitectura de la solución final son los siguientes:

- Tiempo de respuesta (latencia) menor a 10ms.
- Alta disponibilidad (99,999% del tiempo). La caída de un elemento del sistema no puede comprometer el acceso a los datos almacenados en el mismo. No debe existir un punto de fallo único.
- Protocolos soportados: LDAP, JDBC.
- Escalabilidad lineal de todo el sistema manteniendo lineal el CAPEX total.
- Punto de acceso único. El sistema debe tener un punto de acceso único para poder procesar las peticiones LDAP desde los nodos de aplicaciones de la red.
- Capacidad de almacenamiento del sistema de 50 millones de usuarios, con un tamaño medio por usuario de unos 4Kb.

- Inversiones en hardware y software reducidas utilizando tecnología estándar y de código abierto.
- Coste de operación mínimo.

3 Arquitectura del Sistema

A la hora de definir la arquitectura del sistema, el arquitecto se enfrenta a una serie de elecciones, a saber:

- Almacenamiento de datos: memoria RAM, disco magnético o sólido, o combinación de ambos.
- Fragmentación de datos: horizontal o vertical.
- Distribución de los datos: mediante *round-robin*, por *hashing*, por rangos o aleatoria-desigual (*random-unequal*) [7].
- Replicación: síncrona o asíncrona, ROWA/ROWA-A/Copia principal/Quórum [7].
- Gestión de transacciones: protocolos Two-Phase Commit, Three-Phase Commit, ordenación por marca de tiempo (*timestamp ordering*) [8].

El requisito de baja latencia sólo se puede cumplir almacenando los datos en RAM y/o un disco sólido (SSD). En este sentido, la última generación de discos sólidos SSD parece que va a permitir cumplir el requisito de baja latencia, puesto que la generación anterior, probada durante el proyecto, estaba muy cerca de lograrlo.

El patrón de acceso a datos de los nodos que usarán este sistema de BBDD es principalmente búsqueda de una tupla completa por clave primaria o clave secundaria. Por tanto el esquema de fragmentación que mejor se ajusta a la aplicación es el de fragmentación horizontal.

Los requisitos de escalabilidad y bajo OPEX se cumplen más eficazmente con una distribución de datos basada en *hashing*, teniendo en cuenta que no habrá prácticamente búsquedas que no sean por clave primaria/secundaria. El método de *hashing* plantea el problema del potencialmente alto desequilibrio de procesamiento (*processing skew*). Sin embargo al no haber consultas complejas este desequilibrio es asumible por la plataforma de ejecución.

Para resolver el problema de acceso por clave secundaria cuando hay distribución por *hashing* empleamos un esquema BERD [7] para implementar la(s) clave(s) secundaria(s).

Para la replicación se ha optado por un modelo N+K (N fragmentos con K réplicas por fragmento), donde las réplicas se atacan mediante un esquema ROWA-A debido a su mejor respuesta ante fallos individuales. Las réplicas no activas durante una actualización se sincronizan al revivir desde una réplica que esté al día.

El problema de la gestión de transacciones no se ha atacado durante este proyecto y será estudiado en una fase posterior. Probablemente la elección esté entre un 3PC optimizado o un esquema basado en TO, en función del balance eficiencia/fiabilidad que aporte cada uno.

4 Herramientas Utilizadas en el Proyecto

En el proyecto se han utilizado diferentes herramientas de código abierto. La programación se realiza principalmente utilizando el entorno de *gcc* [9] y el lenguaje C++ (debido a su alta eficiencia en ejecución) para las partes críticas del sistema; y Java para los sistemas de gestión y, en general, para las partes del sistema que tienen especiales requisitos de rendimiento. Como entorno de desarrollo se utiliza eclipse [10] con CDT [11]. El sistema operativo sobre el que se ejecuta el sistema es Linux SLES 10.1 (en este caso porque se trata la distribución Linux homologada en Ericsson).

Como plataforma de control de versiones se utiliza CVS [12] y para la programación de compilaciones nocturnas; para la ejecución de *scripts* periódicos y de pruebas funcionales se utiliza HUDSON [13]. Ésta última herramienta ha sido uno de los hallazgos del proyecto, ya que se ha mostrado muy útil y práctica para implementar el proceso de integración continua.

Finalmente, para ejecutar las pruebas de rendimiento en el sistema se ha usado SLAMD así como programas de pruebas hechos a medida. El entorno de ejecución del sistema se compone de las máquinas de la figura 1, con las características ahí expuestas.

	HARDWARE CONFIGURATION				
HOSTNAME	Model	CPU	RAM	Network	Disk
Homer / Marge / Bart / Lisa	DELL Optiplex 755	Core2 Quad 2.66 GHz	8 GB	Intel 82566DM-2 Gigabit Ethernet	250 GB SATA
Philo-A / Philo-B Philo-C / Philo-D	DELL PowerEdge T605	2x Quad-Core AMD Opteron 2350	32 GB	Broadcom NetXtreme Gigabit Ethernet	2x 147 GB SAS
Philo-E / Philo-F Philo-G / Philo-H	DELL PowerEdge T605	2x Quad-Core AMD Opteron 2350	8 GB	Broadcom NetXtreme Gigabit Ethernet	2x 73 GB SAS
Patty / Selma	DELL Optiplex 755	Core2 Duo 3.00 GHz	4 GB	Intel 82566DM-2 Gigabit Ethernet	250 GB SATA

Fig. 1. Equipamiento utilizado

El despliegue del prototipo se describe en la figura 2.

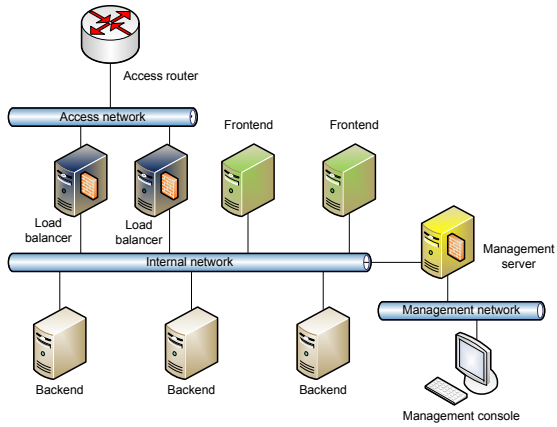


Fig. 2. Configuración de laboratorio para el proyecto

5 Análisis del Sistema

Basados en los requisitos anteriormente expuestos y en el estado de arte existente se propone la siguiente arquitectura multicapa avanzada en la figura 2:

- Punto de acceso único proporcionado generalmente por balanceadores de carga.
- *Front-ends* propios de la aplicación, esto es, que ofrezcan las interfaces típicas de un nodo: LDAP, JDBC, etc.
- Capa de enlace con la base de datos común a todos los nodos
- *Back-ends* comunes que puedan hacer frente a los requisitos que imponen estas aplicaciones.
- Sistema de gestión de la base de datos.

Los *front-ends* y los *back-ends* son las partes centrales del sistema, además del sistema de gestión. Estos dos tipos de nodos son los que centran de una manera más específica el objetivo de este proyecto de investigación.

Los *front-ends* son los elementos encargados de recibir las peticiones desde el cliente externo, procesar la petición generalmente utilizando el protocolo JDBC o LDAP y posteriormente acceder a los *back-ends* para recuperar o insertar la tupla de una forma adecuada. Esta tupla estará replicada en, al menos, dos encarnaciones de *back-ends* para asegurar al menos una disponibilidad 1+1 de cada tupla de la BBDD.

Los *back-ends* son los almacenes de datos propiamente dichos. Estos elementos son los responsables de almacenar la información y de asegurar su consistencia y disponibilidad, así como de asegurar unos tiempos de acceso mínimos que permitan completar cualquier consulta con una latencia apropiada conforme a los requisitos del sistema.

Además de esta estructura de capas podemos ver en la figura 3 las interfaces tanto internas como externas que posee la solución. Estas interfaces permiten la gestión y la

operación de la base de datos distribuida en memoria de una manera totalmente transparente.

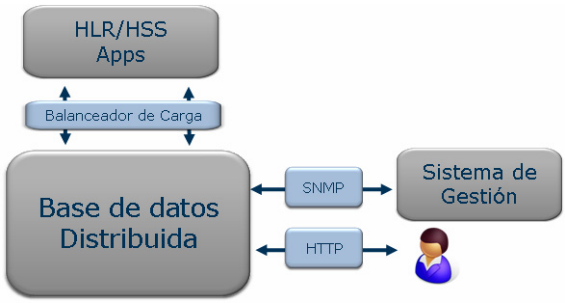


Fig. 3. Interfaces del sistema

Los principios que se siguen en la solución para cumplir con los requisitos planteados al principio del proyecto son los siguientes:

- *Almacenamiento en memoria*: el almacenamiento de los datos en dispositivos magnéticos no es algo asumible ya que los requisitos de latencia no lo permiten. La persistencia se alcanza mediante el almacenamiento de la información en discos magnéticos en forma de REDO *logs*.
- *Redundacia*: las réplicas se reparten las cargas de las consultas.
- *Sistema en capas*: funciones especializadas por capas (acceso y almacenaje). Cada capa escala independientemente. Balanceo de carga eficiente.
- *Principios de diseño del software*: código diseñado para optimizar la velocidad y para ocupar el mínimo de hardware. Software simple y modular. Código portable y abierto.

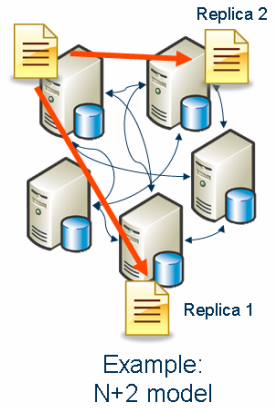


Fig. 4. Despliegue de réplicas en el prototipo

Otro de los puntos importantes del sistema es la distribución de los datos entre los diferentes *back-ends* disponibles(figura 4). Como ya se ha explicado, las tablas se fragmentan horizontalmente y las tuplas se distribuyen entre los *back-ends* activos

mediante un esquema de *hashing*. Cada tupla se registra en al menos dos *back-ends* de modo que siempre haya al menos una réplica de cada tupla en caso de fallo aislado de un *back-end*.

Cuando se producen reconfiguraciones en el sistema, y se añaden o eliminan *back-ends*, habrá tuplas que tengan que moverse de unos *back-ends* a otros. Con un esquema de distribución basado en *hashing*, esto significaría que las tuplas movidas serían ilocalizables tras la reconfiguración. Para evitar este problema, introducimos un nivel adicional de granularidad entre la tabla y la tupla, que llamamos la rodaja (*slice*). Una rodaja es un subconjunto de tuplas, todas ellas pertenecientes a una misma tabla. Cuando hay una reconfiguración no se mueven tuplas individuales sino rodajas completas. Asimismo, introducimos un nivel de indirección en el esquema de *hashing*, de modo que el *hash* de la clave primaria no produce una referencia al *back-end* que contiene la tupla sino a una rodaja. Una tabla distribuida (*Distributed Routing Table*, DRT) contiene la relación entre rodajas y *back-ends*. De este modo, cuando una rodaja se mueve, la DRT es actualizada con su nueva ubicación, y el esquema de *hashing* sigue funcionando para las tuplas desplazadas.

La figura 5 ilustra cómo desde un *front-end* se encuentra el *back-end* donde se halla una tupla determinada por su clave primaria. Cuando una consulta entra por el punto único de acceso al sistema –un balanceador de tráfico– éste le asigna un *front-end* según sea el protocolo que maneje (p. ej. LDAP o JDBC). Una vez que el *front-end* recibe la operación, realiza el proceso de *hashing* al final del cual obtiene referencias a las máquinas donde están ubicadas las réplicas que contienen la tupla buscada. Si es una lectura, se elige una réplica, pero para una actualización se involucran todas las réplicas que estén activas (ROWA-A).

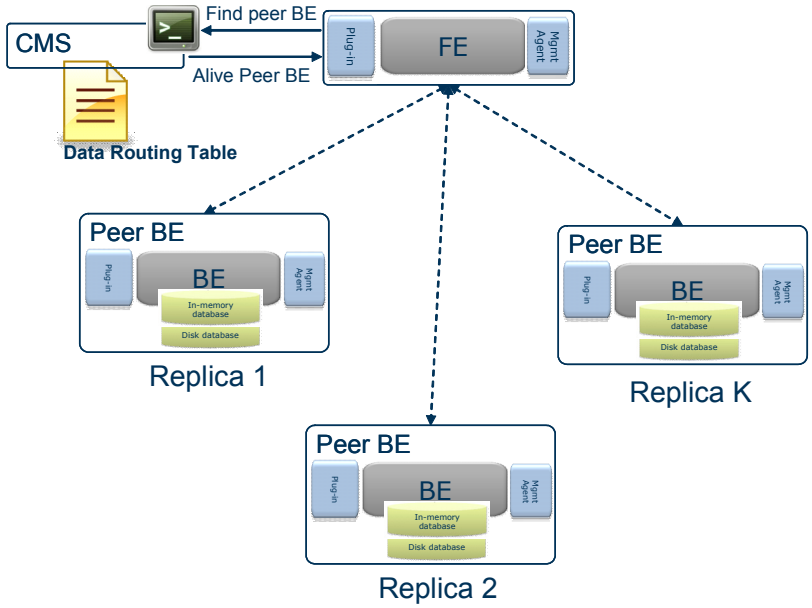


Fig. 5. Proceso de localización de tupla

La figura 6 ilustra la ubicación física de rodajas y réplicas, para un esquema de replicación 1+1. Obsérvese que una tupla y su réplica siempre se ubican en máquinas diferentes para aumentar la probabilidad de supervivencia en caso de fallo aislado de una máquina.

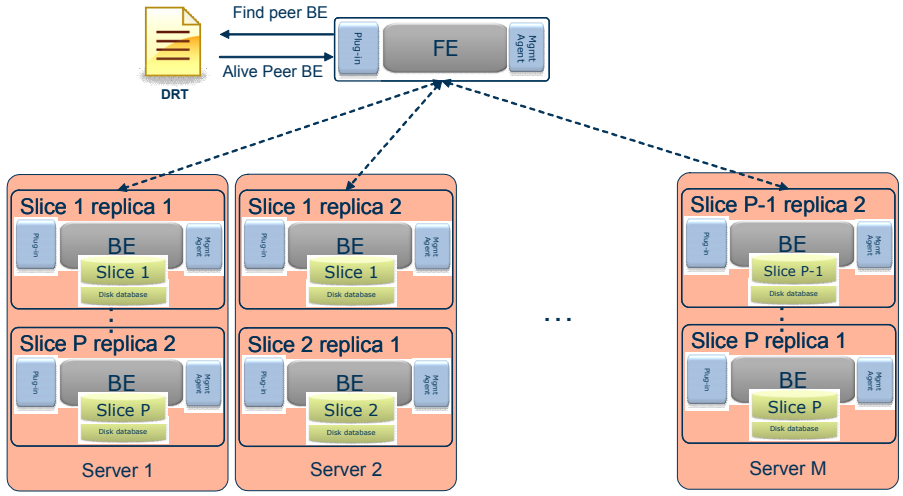


Fig. 6. Distribución de tuplas en los *back-ends* por rodajas

La figura 7 ejemplifica la estructura de la DRT. La columna “Range” en la figura contiene identificadores de rodaja. Las columnas “Peer” contienen referencias a las

máquinas donde están físicamente ubicadas las rodajas (réplicas) de cada identificador de rodaja.

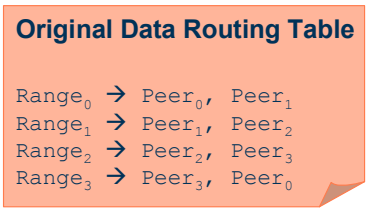


Fig. 7. Tabla de enrutamiento de datos

La escalabilidad de la solución se asegura con la redistribución de la información entre las rodajas existentes y las nuevas que se puedan generar.

Una de las características a resaltar del sistema es su completa modularidad permitiendo en todo momento la sustitución de cualquier elemento por otro permitiendo de esta manera la utilización del prototipo como banco de pruebas de nuevas soluciones en bases de datos distribuidas en memoria.

6 Resultado de las Pruebas

Los resultados de las pruebas efectuadas en el sistema son muy prometedores en relación a los requisitos impuestos inicialmente. Los parámetros que han sido objeto de medición en este sistema inicial son los de tiempo real, capacidad, *throughput*, disponibilidad, escalabilidad lineal, punto de acceso único y CAPEX reducido. Estos parámetros, inicialmente contemplados como básicos en los requisitos iniciales del sistema, han sido medidos utilizando como herramienta principal SLAMD, descrita en el apartado de herramientas. Partiendo de la configuración ilustrada en las figuras 1 y 2, se procedió a inyectar tráfico LDAP por medio de generadores SLAMD. Cada cliente SLAMD permite, a su vez, configurar el número necesario de instancias generadoras de tráfico LDAP que, funcionando de manera síncrona, lleve al sistema al punto de saturación.
Se han realizado pruebas para cada elemento del sistema por separado (LB, FE, BE), así como del sistema completo, llegando en ambos casos a consecuencias similares.
El tráfico generado seguía una distribución uniforme de forma que cada elemento del sistema de pruebas recibía una intensidad de tráfico similar.
El tamaño medio de los datos almacenados en los BE era de 4KB y el patrón de tráfico era de 60% de lecturas y 40% de escrituras (aunque también se hicieron algunas pruebas para cada tipo de operación LDAP por separado).
Para cada prueba se obtuvieron las curvas de carga de CPU, latencia, throughput y memoria consumida.
Las conclusiones fundamentales de las pruebas realizadas son las siguientes:

- *Tiempo real*: el sistema se ha comportado de forma excelente para cargas de unos 300K usuarios (tuplas) de un tamaño medio de 4kb por registro. Se han alcanzado

tiempos de respuesta que se encuentran dentro del rango de 10 ms para ciclos completos de lectura-escritura a través de todas las capas del sistema. La arquitectura del sistema, como ya se ha visto, permite que el tiempo de respuesta varíe de forma logarítmica con respecto al número de tuplas, lo que permite extrapolar que los tiempos de respuesta se mantendrían dentro del margen requerido aunque aumentase significativamente el número de éstas.

- *Escalabilidad*: la escalabilidad, medida como la capacidad para incrementar el número de máquinas sin afectar significativamente otros parámetros del sistema, ha sido comprobada para un número de máquinas reducido. Al no existir ningún tipo de comunicación entre los servidores de *back-end*, pueden ser añadidos tantos *back-ends* como sean necesarios sin que aumente la tara debida a la comunicación entre ellos.
- La *capacidad* del sistema ha sido comprobada de una manera extrapolada. Se ha verificado que el sistema es completamente escalable en cuanto a número de usuarios y número de máquinas y por tanto puede deducirse que la capacidad del sistema puede extenderse a millones de usuarios sin presentar limitaciones importantes.
- *Throughput*: se han realizado pruebas de carga usando múltiples configuraciones de servidores de *back-end*: "share-nothing" (un único proceso por servidor) y "share-something" (múltiples procesos por servidor, tantos como núcleos de CPU, compartiendo interfaz de red, acceso a memoria y disco magnético), con distintos umbrales de uso de CPU (20%, 50%, 80%). Se ha verificado que los tiempos de respuesta se mantienen dentro de los límites requeridos incluso en el caso de 80% de carga de CPU, si bien en las pruebas sintéticas que se han llevado a cabo no se ha introducido disparidad ("skew"), es decir, todas las CPUs soportaban aproximadamente el mismo nivel de carga.
- *Alta disponibilidad*: no se han realizado pruebas en esta fase de prototipado que puedan demostrar que el sistema tiene 5 nueves de alta disponibilidad.
- *Punto de acceso único*: gracias a los balanceadores de carga, el sistema es capaz de repartir las consultas de forma homogénea entre todos los servidores de *front-end* disponibles, haciendo un aprovechamiento óptimo de los recursos. Todas las consultas tienen un punto de acceso único al sistema. Aunque el software de equilibrado de carga que se ha empleado soporta un reparto más "inteligente" de las consultas, p. ej., basándose en el nivel de carga de cada servidor, como ya se ha dicho no se han hecho pruebas con disparidad ("skew").
- *CAPEX reducido*: mediante el uso de recursos de computación estándar, y gracias a las características especiales de la arquitectura, se asegura que las inversiones son linealmente proporcionales al número de usuarios que son necesarios almacenar en el sistema, asegurando la escalabilidad lineal de la solución.

7 Conclusiones

El prototipo inicial desarrollado proporciona una primera versión experimental de una base de datos distribuida en memoria que cumple con los requisitos establecidos por los nodos de las redes de telecomunicaciones que manejan la información de los

usuarios. Es especialmente destacable el alto grado de robustez alcanzado por la solución. Este primer prototipo demuestra la viabilidad del desarrollo de este tipo de soluciones basadas en herramientas de desarrollo de código abierto y en plataformas totalmente estándares.

Los resultados anteriormente descritos muestran una primera fase inicial en la evolución de un prototipo de base de datos en memoria. El proyecto anteriormente comentado se encuentra en fase de investigación y generará en el futuro más resultados con la incorporación de nuevas tecnologías.

Agradecimientos

Este proyecto ha sido desarrollado en la Unidad de Tecnología e Innovación del Centro de I+D de Ericsson España con la colaboración del Departamento de Ingeniería Telemática (DIT) y del Laboratorio de Sistemas Distribuidos de la Facultad de Informática, ambos pertenecientes a la Universidad Politécnica de Madrid.

References

1. Oracle 11g, http://www.oracle.com/database/index.html
2. MySQL Enterprise, http://www.mysql.com/products/enterprise/
3. Xeround Intelligent Data Grid Suite, http://www.xeround.com/415-xeround-intelligent-data-grid-suite
4. Nokia Siemens Networks Subscriber Data Management, antiguo Apertio, http://www.nokiasiemensnetworks.com/global/Portfolio/Products/Core/Subscriber+Data+Management/Subscriber+Data+Managemenent.htm
5. MySQL Cluster, http://www.mysql.com/products/database/cluster/
6. Stonebraker, M., Madden, S., Abadi, D. J., Harizopoulos, S., Hachem, N., and Helland, P. 2007. The end of an architectural era: (it's time for a complete rewrite): Proceedings of the 33rd international Conference on Very Large Data Bases (Vienna, Austria, September 23 - 27, 2007). Very Large Data Bases. VLDB Endowment, 1150-1160.
7. D. Taniar, C. Leung, W. Rahayu, S. Goel: High-performance Parallel Database Processing and Gris Databases, Wiley 2008
8. P. Bernstein, N. Goodman: Concurrency Control in Distributed Database Systems, Computing Surveys Vol.13 No. 2 1981
9. GCC, the GNU Compiler Collection, http://gcc.gnu.org
10. Eclipse, http://www.eclipse.org
11. Eclipse C/C++ Development Toolkit, http://www.eclipse.org/cdt
12. cvs – Concurrent Versions System, http://www.nongnu.org/cvs/
13. hudson: an extensible continuous integration engine, https://hudson.dev.java.net/

Microsistemas de Información[†]

Jordi Pradel[1,2], Jose Raya[1,2], Xavier Franch[2]

[1]Agilogy
Av. Meridiana, 519 1r, 08016 Barcelona, Spain
{jordi.pradel, jose.raya}@agilogy.com
[2]Universitat Politècnica de Catalunya (UPC)
c/Jordi Girona, 1-3, E-08034 Barcelona, Spain
{jpradel, jraya, franch}@lsi.upc.edu

Resumen. Ante la necesidad de gestionar la información que manejan, las organizaciones disponen de dos tipos de herramientas informáticas ampliamente utilizadas: los sistemas de información basados en bases de datos (SIBD) que resultan costosos de desarrollar y son poco flexibles; y las hojas de cálculo (HdC), que ponen en peligro la integridad de los datos y son limitadas en cuanto a la explotación de los mismos. En el presente artículo proponemos un nuevo paradigma intermedio, los microsistemas de información (MicroIS), que permiten fluctuar, de la manera más automatizada posible, entre la flexibilidad de las HdC y la integridad de los SIBD obteniendo, de este modo, lo mejor de ambos mundos: el bajo coste de desarrollo y mantenimiento, la facilidad de uso y la flexibilidad de una HdC; y la estructura, la semántica y la integridad de un SIBD. El objetivo no es reemplazar ninguno de los dos paradigmas anteriormente mencionados sino situarse en un punto intermedio según evolucionen las necesidades de la organización. De los diversos puntos de interés de este paradigma, el artículo se centrará precisamente en los aspectos relacionados con la estructuración de los datos, presentando el modelo conceptual de los MicroIS, las transformaciones y validaciones que pueden realizarse sobre el mismo, y la forma en que se infiere la estructura de la información a partir de los datos que entra el usuario.

Palabras clave: sistema de información, hoja de cálculo, microsistema de información, auto-organización de datos.

1 Introducción

Hoy en día, las organizaciones requieren el almacenado y proceso altamente automatizado de grandes volúmenes de información de todo tipo. Para ello, disponen de diversas opciones. Por un lado, el desarrollo de la tecnología de los sistemas de información permite gestionar la información de forma eficiente y segura con propósitos tales como el procesado de transacciones, el soporte a la toma de decisiones, la gestión del conocimiento, etc. Dichos sistemas presentan normalmente una arquitectura apoyada en una base de datos (Sistemas de Información basados en

[†] Este trabajo ha sido parcialmente apoyado por el proyecto TIN2007-64753.

Bases de Datos, SIBD) típicamente relacional a la que se añaden funcionalidades y capacidades mediante software desarrollado a medida o adquirido/licenciado. En el caso extremo, el sistema se construye mediante un proceso de parametrización e integración de uno o más sistemas comerciales como sistemas ERP, CRM o DM.

Otro tipo de herramienta que ha facilitado el procesado de información en las organizaciones son las hojas de cálculo (HdC). Mediante un paradigma sencillo de entender (cada hoja contiene una tabla organizada en filas y columnas) y ofreciendo una flexibilidad prácticamente absoluta en cuanto a los contenidos de cada celda de la tabla, se facilita enormemente la creación, con un coste mínimo, de pequeños sistemas que permitan almacenar la información y solucionar, aunque sea parcialmente, la necesidad de automatización sin necesidad de tener conocimientos sobre desarrollo.

Ambos paradigmas se sitúan en extremos opuestos en cuanto a flexibilidad, ya que mientras los SIBD se caracterizan por restringir las entradas de datos válidas, las HdC permiten introducir prácticamente cualquier valor en cualquier celda. Esta divergencia se traduce en una enorme diferencia en cuanto a la integridad de los datos, que es total en el caso de los SIBD y prácticamente inexistente en el caso de las HdC. También se sitúan en extremos opuestos en cuanto al coste de desarrollo puesto que las HdC permiten aprovechar una misma interfaz de usuario genérica para todas las aplicaciones mientras que en los SIBD son necesarios o bien el desarrollo de un sistema completo, o bien la parametrización de un sistema genérico, siendo en ambos casos necesarios conocimientos de desarrollo e integración, lo que conlleva un elevado coste inicial en el que hay que incurrir antes de poder disfrutar de sus beneficios. Por último, otro aspecto igualmente importante es la diferencia por lo que a movilidad se refiere: mientras las HdC permiten su fácil traslado de un entorno a otro sin necesidad de instalación y/o configuración (o incluso directamente su uso compartido a través de Internet), los SIBD precisan de un proceso de migración no tan inmediato. La Tabla 1 resume la manera en que los SIBD y las HdC satisfacen las características mencionadas y otras igualmente importantes.

Tabla 1. Comparativa entre SIBD y HdC.

Característica	*HdC*	*SIBD*
Estructura de la información flexible	+	-
Posibilidad de estructurar la información y dotarla de semántica	-	+
Manejo de grandes volúmenes de datos	-	+
Retorno de la inversión rápido	+	-
Movilidad	+	-
Posibilidad de compartir entre distintos usuarios sin proliferación de copias	-	+
Posibilidad de personalizar consultas / vistas sin programar	+	-
Consultas potentes (más allá de filtros y ordenación)	-	+
Vistas maestro/detalle	-	+
Varias vistas sobre el mismo conjunto de datos	-	+
Creación de gráficos	+	-
Asociaciones: Relacionar elementos	-	+
Gestión por parte de TI (copias de seguridad, control de acceso, versionado, etc.)	-	+
Reglas de integridad	-	+

El propósito de este artículo consiste en presentar una solución que permite construir sistemas de información con las mismas propiedades que las HdC: facilidad de uso, poca inversión y resultados inmediatos pero que, a diferencia de las mismas, puede evolucionar hacia algo más parecido a un SIBD, con reglas de integridad, relaciones

entre datos, etc. Dado que nuestro objetivo es facilitar la construcción de pequeños sistemas de información, hemos dado a llamar, al nuevo paradigma, **microsistemas de información** (abreviadamente, MicroIS). La Figura 1 resume la razón de ser de los MicroIS como compromiso entre los SIBD y las HdC.

Como puede deducirse de esta introducción la problemática de los MicroIS es compleja y por motivos de espacio nos centraremos en el presente trabajo en aspectos directamente relacionados con el tema de estructuración de los datos:

- Modelado implícito asistido: El sistema se encargará, mediante heurísticas adecuadas, de sugerir al usuario posibles mejoras en el esquema de datos a partir del estudio de la información almacenada, de modo que un usuario sin conocimientos de modelado conceptual pueda añadir estructura a su sistema de información. De este modo el esquema de datos podrá evolucionar de forma segura y con facilidad, sin afectar a los datos ya guardados.
- Reglas de integridad no intrusivas: Usando la metáfora del corrector ortográfico de una herramienta ofimática: el sistema conocerá las reglas de integridad y mostrará al usuario aquellos valores que no las cumplan e incluso sugerencias para su corrección cuando las tenga, pero en ningún momento impedirá al usuario introducir los datos. Más adelante, el usuario podrá localizar las inconsistencias y corregirlas si así lo desea.

Otros aspectos de los MicroIS igualmente importantes, como el ya citado de la movilidad, quedan pues fuera del ámbito de este artículo.

2 Trabajo Relacionado

La dicotomía SIBD – HdC es bien conocida y ya ha sido abordada con anterioridad por diversos autores. Por ejemplo, en [1] el autor establece la problemática de los SIBD cuya utilidad no garantiza el retorno de la inversión exigida por su desarrollo y lo relaciona con el concepto de "long-tail" [2] al asumir que existe un número enorme de pequeños sistemas de información que no han llegado a ser desarrollados como tales por ese motivo. La conclusión a la que llega es que, en cualquier caso, la solución no es utilizar HdC.

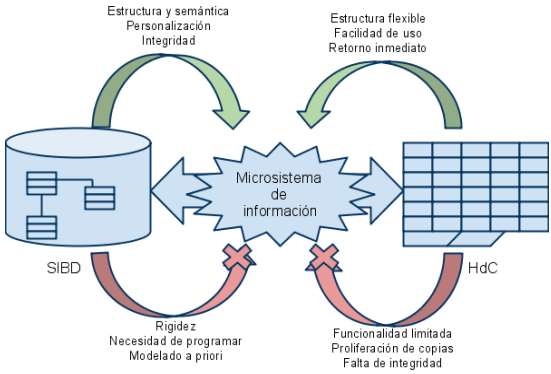


Fig. 1. Los MicroIS como un compromiso entre los SIBD y las HdC.

¿Por qué, entonces, se utilizan las HdC para implementar este tipo de sistemas? Creemos que una de las claves la podemos encontrar en [3] donde, sobre una población de aproximadamente 1600 individuos, se analiza el uso que éstos hacen de las hojas de cálculo, observándose que tan sólo el 23% de los encuestados realizaba algún tipo de modelado previo al desarrollo de la HdC. La no necesidad de modelado a priori facilita la etapa de creación ya que el usuario puede introducir los datos en el momento inicial, y añadir las fórmulas y formatos a medida que se vaya detectando su necesidad. Los problemas suelen aparecer a medida que las necesidades de información evolucionan y empiezan a darse las primeras inconsistencias o, sencillamente, el uso diario del mismo se vuelve farragoso dada la poca ayuda por parte del sistema para la introducción y explotación de los datos (al contrario que los SIBD, que normalmente presentan una capa de presentación diseñada con ese propósito). Llegados a este punto, la solución de la HdC no puede evolucionar más y la organización vuelve a encontrarse ante la necesidad de abordar el coste del desarrollo de un sistema de información. Por ello, el autor de [1] acabó desarrollando la aplicación DabbleDB [4], con la intención de facilitar la manipulación de los datos y la evolución del esquema. A pesar de compartir las conclusiones, creemos que DabbleDB no es la opción ideal para este tipo de sistemas ya que requiere una conexión activa con un servidor en Internet y, por lo tanto, no es utilizable en cualquier entorno. Además, carece de la capacidad de autoorganizarse, por lo que toda la responsabilidad de modelado recae sobre el usuario sin ayuda por parte del sistema.

Una solución alternativa sería el tratamiento relacional de la información almacenada en una HdC. Así, en [5] se presenta la herramienta HaExcel, que propone una correspondencia bidireccional entre HdC y bases de datos relacionales (BDR). No obstante, los objetivos mencionados en la introducción, modelado implícito asistido y reglas no intrusivas, no forman parte de los objetivos del proyecto. En Microsoft ADO.net [6] también se contempla la posibilidad de establecer esta correspondencia a costa de limitar significativamente la estructura de partida de la HdC.

Otra aproximación al problema consiste en reducir el coste del desarrollo del sistema de información mediante la generación automática de aplicaciones, basada en modelos como en AndroMDA [7], en código como en OpenXava [8] o en bases de datos existentes como en Ruby On Rails [9]. A pesar de sus conocidas ventajas, consideramos que todas estas herramientas comparten una serie de inconvenientes derivados del uso de una BDR: la necesidad de modelado a priori, que impide a los usuarios sin conocimientos de desarrollo implementar sus propias soluciones (por lo que no consiguen desplazar a las HdC como soluciones "pragmáticas"); la falta de flexibilidad en cuanto al cumplimiento del esquema; y la dificultad de evolución, puesto que el proceso de generación de la aplicación es unidireccional.

3 Caso de Estudio

A modo ilustrativo presentamos una versión simplificada del proyecto desarrollado por la empresa Agilogy en la que trabajan dos de los autores (v. [10]). Se trata de una aplicación que gestiona información sobre los datos antropométricos y las visitas de los clientes de una consulta de dietética. Dicha aplicación fue desarrollada para una empresa (llamada Lalinia) que, dado su pequeño tamaño, no podía permitirse el

desarrollo y puesta en explotación de un SIBD debido al alto coste de dicha solución pero que, sin embargo, no quería utilizar una HdC por los riesgos relacionados con la falta de integridad de los datos. Los factores decisivos para elegir la opción de desarrollar un MicroIS fueron:

- Pequeño volumen de información: El volumen de datos a tratar era lo suficientemente manejable como para considerar viable el uso de MicroIS.
- Bajo coste de desarrollo: Una vez desarrollada la aplicación ofimática para la gestión de MicroIS, crear un MicroIS concreto como el de Lalinia tendría un coste muy bajo.
- Flexibilidad: Al tratarse de un desarrollo nuevo, Lalinia no estaba seguro de poder acertar los requisitos antes de utilizar el sistema en producción durante un tiempo; incluso, la estabilidad de los requisitos de datos a lo largo del tiempo no estaba garantizada. Por ello, se consideró fundamental poder ir añadiendo estructura a medida que se fuesen detectando nuevas necesidades.
- Facilidad de uso: Era importante que la herramienta no interfiriese con el curso natural de la visita de los clientes, por lo que Lalinia estaba dispuesto a permitir entradas con errores que se corregirían más adelante, una vez finalizada la visita del cliente o que incluso podrían quedar sin resolver.
- Ayudas a la introducción de datos: El sistema debería tener conocimiento sobre la semántica de los datos para poder facilitar la introducción de los mismos (escoger los valores de una asociación desde una lista desplegable, ayudar a cumplir las restricciones de integridad asociadas a un atributo, etc.).

En la Figura 2 se muestra un posible escenario que ilustra los factores mencionados. Se representan tabularmente los dos grandes tipos de información a almacenar: la información personal y los datos antropométricos; en la segunda hoja se representa cada visita por una fila (ítem). Se pueden advertir algunas de las situaciones típicas de estos MicroIS referentes a la estructuración de los datos:

- Atributos específicos de un ítem: Por ejemplo, en este caso, hay un ítem (Luisa Fernández) que tiene un dato (la dirección de la casa de veraneo) que, normalmente, no tendremos. Dichos datos no son conocidos a priori, es decir, cada cliente puede suministrar ciertos datos particulares y Lalinia quiere ser capaz de mantener clasificada la información que considere útil en lugar de simplemente guardarla en un campo genérico de comentarios.
- Documento con errores potenciales: Podría ser que el paciente Juan Pérez no hubiese suministrado todavía un dato obligatorio como su teléfono y, no obstante, Lalinia requiere que se pueda almacenar el resto de datos del ítem a la espera de obtener tal información.
- Poca estructuración: Mientras que el peso es un dato que se mide en todas las visitas, la altura podría no medirse siempre ya que las variaciones en altura son mucho menores que las variaciones en peso.
- Alta variabilidad. Podemos observar las diferentes formas en que los clientes suministran el teléfono de contacto. Desde un simple número sin más comentarios, hasta diversos números, extensión, horarios, tipo, etc. Cualquier intento de anticiparse a todos los casos y dotar de estructura a priori está condenado al fracaso (no es difícil imaginarse algunos casos más que los presentados en la figura).

Información Personal						
Nombre	Apellido	Teléfono	e-mail	Fax	Dirección	Dirección de veraneo
Jordi	Garcia Gil	934130000 ext. 512		912354875	c/Mayo, 1	
Montse	Tolrà	934444444 10-14h 666666666 (móvil)	mtolra@gmail.com		c/Jardín,5	
Marta	Vallès	45665432 mañanas 99999999 tardes		932584875	c/Altar,65	
Juan	Pérez López		juan@example.com	931234567		
Luisa	Fernández Martínez	935559876	luisa@example.com		c/Pí, 314	Urbanización Las Torres, 5

Datos Antropométricos				
Nombre	Apellido	Fecha	Peso	Altura
Juan	Pérez López	15/5/09	120 kg	185 cm
Jusn	Pérez López	20/5/09	115 kg	
Luisa	Fernández Martínez	17/5/09	50 kg	160 cm
Luisa	Fernández Martínez	25/5/09	52 kg	1'60

Fig. 2. Un posible estado del MicroIS para Lalinia.

La solución propuesta permite convivir con estas situaciones pero al mismo tiempo provoca una cierta vulnerabilidad ligada a la posible existencia de inconsistencias. Dichas inconsistencias pueden ser a nivel intra-hoja (por ejemplo, en la hoja de Datos Antropométricos se puede observar un error tipográfico en la introducción del nombre en la segunda fila) o inter-hoja (por ejemplo, en esa misma hoja el nombre y apellidos del paciente se copian manualmente, por lo que si se modifica en la primera hoja -e.g. hubo un error de transcripción en la primera visita- no se modifica automáticamente la segunda). Dichas situaciones deben ser consideradas también en la solución dada.

Por parte de Agilogy, se había trabajado a nivel teórico sobre el paradigma pero no se disponía de ninguna aplicación operativa por lo que se decidió utilizar la aplicación de Lalinia como prueba piloto que dirigiría el desarrollo de una versión preliminar de la herramienta de gestión de los MicroIS. La implementación del MicroIS ya ha terminado de forma satisfactoria, habiéndose entrado actualmente en una fase de seguimiento que permitirá próximamente extraer unas primeras conclusiones sobre el paradigma.

4 Modelo Conceptual de Datos de los MicroIS

Un MicroIS es, en el fondo, un pequeño repositorio de datos. Sin embargo, su naturaleza es más cercana al documento ofimático que a la BDR típica basada en un servidor centralizado ya que se trata de manejar volúmenes no muy grandes de información mediante una herramienta interactiva que gestiona, al mismo tiempo, la estructura, los datos y la interfaz gráfica. Dicha interfaz gráfica se basará en un modelo genérico para todas las aplicaciones aunque, al igual que las herramientas ofimáticas, permitirá su personalización.

Si nos centramos en la perspectiva de los datos, la diferencia fundamental entre un MicroIS y una BDR es la independencia de los datos almacenados respecto al esquema que determina su estructura, al estilo del patrón "Type Object Pattern" [11].

Ello se refleja en el modelo conceptual de datos en UML que se presenta en la Figura 3, donde aparecen dos zonas claramente diferenciadas a izquierda (datos) y derecha (esquema) de la clase MicroIS[1].

Como los ítems de datos existen independientemente de las clasificaciones, serán identificados individualmente mediante un identificador artificial generado por el sistema. Cada ítem tendrá asociadas un conjunto de propiedades (identificadas por el nombre dentro del ítem, p.e. Apellido o Dirección de Veraneo) compuestas por un conjunto ordenado de valores (p.e., los diversos valores de Teléfono en aquellos ítems que tienen más de uno). Dichos valores podrán ser valores primitivos (números, textos, fechas, etc.), referencias a otros ítem o fórmulas.

Por su parte, el sistema de tipos que define el esquema de un MicroIS está formado por tipos primitivos (soportado de manera nativa por el sistema) y clases. Las clases son definiciones genéricas de la estructura de un ítem (p.e., Información Personal y Datos Antropométricos) y, como tales, definen un conjunto de atributos que serán de un tipo concreto: tipo primitivo, fórmula o referencia (bidireccional, por lo que será necesario conocer el atributo inverso, v. asociación *inverse* en la figura). Las clases permiten aportar consistencia al sistema al mismo tiempo que facilitan la introducción de datos por parte del usuario. También facilitarán la creación de vistas, fórmulas y consultas al proporcionar un método de selección de ítems.

Cada ítem puede ser clasificado como perteneciente a una clase del esquema. Dicha clase actuará como una plantilla de las propiedades que se espera que tenga el ítem (cada atributo se corresponderá con la propiedad del ítem que tiene su mismo nombre), además de establecer un conjunto de reglas de validación (ver Sección 5).

Como resumen, podemos constatar que este modelo conceptual cumple los requisitos establecidos en las secciones anteriores sobre los MicroIS:

- Flexibilidad. Cada ítem describe un conjunto arbitrario de propiedades, sin restricciones, por lo que dos ítems de una misma hoja pueden diferir en cuanto a las propiedades que contienen y en su contenido (en tipo y/o multiplicidad).

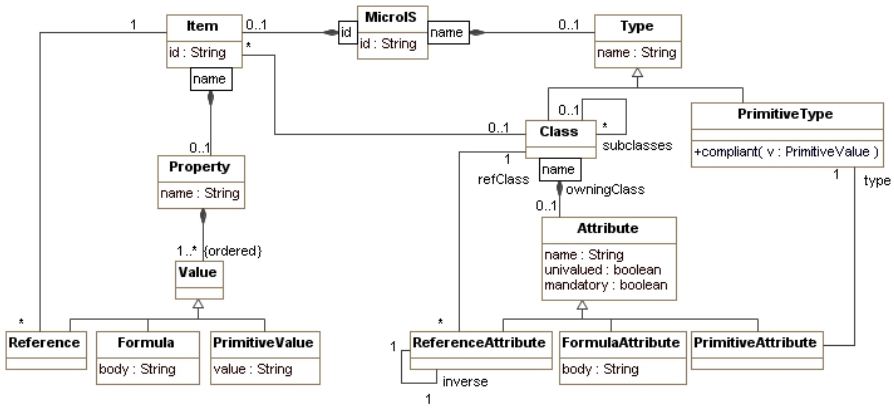


Fig. 3. Un modelo conceptual de datos en UML para los MicroIS.

[1] Por motivos de espacio, no presentamos la totalidad del modelo y en concreto, se omiten las restricciones de integridad (invariantes expresados en OCL).

- Estructura. Al mismo tiempo, para aquellas partes de ítem que son concordantes con el esquema actualmente disponible, se mantiene una correspondencia explícita por nombre.
- Consistencia. La existencia de referencias como valores de propiedades permite conectar ítems y asegurar la consistencia especialmente en el caso inter-hoja.

5 Validaciones

Como se ha comentado, se propone que los MicroIS lleven a cabo ciertas validaciones de datos más allá de las simples validaciones soportadas por las HdC. Dichas validaciones forman parte del esquema y, por lo tanto, se definen junto con éste y su cumplimiento se verifica de la misma manera que el resto de restricciones del esquema; de hecho, el tipo de validaciones que deberá cumplir un ítem dependerá de sus clases y atributos. Todas las restricciones están registradas a nivel del MicroIS mientras que un ítem debería cumplir todas aquellas restricciones que le sean aplicables: una herramienta que implemente el paradigma propuesto deberá, por lo tanto, verificar el cumplimiento del esquema y las validaciones para los diferentes items. El mecanismo de validaciones se ha definido extensible para que, en un futuro, sea posible añadir en los MicroIS nuevos tipos de restricción de integridad (por ejemplo, restricciones de clave) e, incluso, permitir que los usuarios puedan añadir sus propias restricciones.

El subsistema de validaciones se ha integrado en el modelo conceptual de datos de los MicroIS, tal y como puede verse en la Figura 4, en la que se han incluido las restricciones implementadas actualmente.

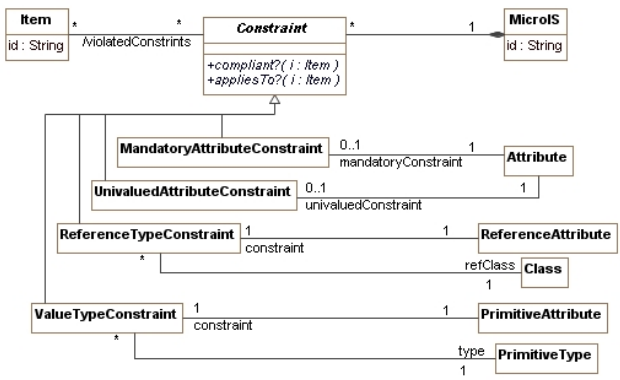


Fig. 4. Un modelo conceptual de datos en UML para las validaciones en los MicroIS.

Puede observarse que la clase Constraint ofrece dos operaciones sobre Items, `Constraint::appliesTo?(i: Item)` indica si una Constraint debe ser comprobada para un determinado ítem mientras que `Constraint::compliant?(i: Item)` indica si un ítem para el que se deba comprobar una Constraint es compatible con ésta o, por el

contrario, la viola. Dichas operaciones son abstractas, por lo que deben definirse en cada tipo de restricción definido. Como ejemplo, la Fig. 5 muestra la definición de tales operaciones para dos Constraint concretas. Ambas restricciones son aplicables a ítems cuya clase corresponda con la clase del atributo asociado a la Constraint.

```
context MandatoryAttributeConstraint::appliesTo?(i:Item) =
                            i.class=self.attribute.class
context MandatoryAttributeConstraint::compliant?(i:Item) =
                           i.property[self.attribute.name]->size()>0
context ValueTypeConstraint::appliesTo?(i:Item) =
                           i.class=self.primitiveAttribute.class
context ValueTypeConstraint::compliant?(i:Item) =
     (i.property[self.primitiveAttribute.name]->size()==0 or
      i.property[self.primitiveAttribute.name]->
            foreach(v|v.oclIsTypeOf(PrimitiveValue) and
                      self.primitiveAttribute.primitiveType.compliant?(v)))
```

Fig. 5. Definición en OCL de dos tipos de restricciones (Constraints).

6 Transformaciones

Desde el punto de vista de la estructura de los datos, la principal característica de los MicroIS es la flexibilidad del esquema. Para mantener la integridad de la información es necesario definir con exactitud qué transformaciones se pueden llevar a cabo sobre el esquema y cómo afectarán tanto al propio esquema como a la población de ítems actuales. Para estudiar dicho problema nos basamos, parcialmente, en el trabajo de Ambler [12], en el que estudia un conjunto de refactorings para introducir cambios en BDR que ya están en producción y propone una serie de técnicas para migrar progresivamente los sistemas que utilizan dichas BDR. A diferencia del refactoring en BDR, modificar el esquema es mucho más fácil cuando los ítems existen fuera de él y el cumplimiento del mismo es opcional y, por lo tanto, se pueden permitir situaciones temporales en que no se cumplan todas las restricciones.

Hasta el momento, hemos identificado 16 transformaciones atómicas que hemos implementado en esta primera versión de los MicroIS: Clasificar/Desclasificar Ítem, Crear Clase a partir de Ítem, Añadir/Quitar Atributo a Clase, Añadir Atributo desde Propiedad, Convertir Atributo Opcional en Obligatorio (y viceversa), Renombrar Atributo, Cambiar Tipo de Atributo Primitivo (p.e., de texto a número), Cambiar Atributo Primitivo a Referencia (y viceversa), Cambiar Atributo a Fórmula, Mover Atributo de Clase, y Crear Ítem desde Valor. También es posible la existencia de transformaciones compuestas. Por ejemplo, Crear Clase a partir de Valor se definiría como Crear Ítem desde Valor seguida por Crear Clase a partir de Ítem.

Como ejemplo, en la Fig. 6 se muestra el detalle de una de ellas, la de clasificación de ítems, usando una plantilla que hemos definido a tal efecto.

Nombre: Clasificar Ítem

Resumen: Añade una clase C a un ítem I

Motivación: Indicar que el ítem I es de clase C para que, en adelante, se apliquen las reglas de validación de C sobre I, y se incluya a I en las vistas y consultas de ítems de clase C

Efectos sobre los datos: Ninguno

Efectos sobre el esquema: el ítem I incluye la clase C en el conjunto de sus clases

Restricciones que pueden pasar a ser violadas: Todas

Restricciones que pueden dejar de ser violadas: Las restricciones asociadas a la clase anterior de I, si la hubiere

Contrato: -- descripción en OCL de los efectos de la transformación sobre el modelo

```
context MicroIS::classifyItem(itemId: String, className: String)
    let I: Item = self.items[itemId] in
    let C: Class = self.types[className] in
    pre: I->notEmpty()
    pre: C->notEmpty()
    post: I.class = C
```

Fig. 6. Definición de la transformación Clasificar Ítem.

7 Autoestructuración: Modelado Implícito Asistido

En la sección anterior hemos presentado las transformaciones que hemos identificado sobre los MicroIS. La siguiente cuestión es cuándo deben aplicarse. Si bien podría pensarse en que el usuario del MicroIS podría invocarlas cuando lo considerara necesario, precisamente hemos identificado como un requisito de los MicroIS evitar que el usuario deba de tomar proactivamente estas decisiones.

Por lo tanto, la opción que contemplamos es que los MicroIS, del mismo modo que sugieren correcciones a los problemas de validación, sugieran mejoras en la estructuración de la información. Es lo que hemos dado a llamar **modelado implícito asistido** ya que el usuario realizaría las tareas de modelado como parte del mantenimiento general de la información (siempre bajo la batuta del MicroIS) y no como una actividad separada. En otras palabras, se facilita la estructuración del MicroIS por parte de usuarios sin conocimientos específicos de modelado de datos ya que ésta se podría llevar a cabo mediante preguntas simples del tipo "¿es este ítem también una visita?" o "¿todos estos datos pertenecen a la misma clase?".

El modelado implícito podrá llevarse a cabo en dos situaciones diferentes:

- En respuesta a un evento externo. Por ejemplo, al dar de alta un ítem, puede crearse un nuevo atributo para la clase a la que pertenece. Si consideramos el ejemplo de la Fig. 2, al dar de alta la última fila de la clase Datos Personales, el sistema podría sugerir la creación del atributo "Dirección de veraneo", que el usuario podría aceptar o no. Igualmente, a medida que se fueran dando de alta ítems con diferentes valores de la propiedad "Teléfono", el sistema podría sugerir afinar la definición del atributo correspondiente en el esquema (multiplicidad, existencia de extensión, horarios de localización, etc.).
- Al cumplirse una condición en la población de ítems del modelo del MicroIS. Por ejemplo, el MicroIS podría analizar los ítems de una clase y descubrir que el porcentaje de los mismos que tiene la propiedad

correspondiente al atributo "Fax" está por debajo de cierto umbral. En este caso, sugeriría al usuario la posibilidad de aplicar la transformación "Quitar atributo a clase" sobre el esquema. Otro ejemplo sería el de crear una clase a partir de un conjunto de ítems; en este caso habría que determinar qué combinación de atributos refleja la relación óptima entre el número de ítems incluidos en la clase y la similitud entre ellos.

No todas las sugerencias deben ir necesariamente dirigidas a la identificación de clases o atributos; también podríamos tener en cuenta la detección de ítems duplicados o la detección de tipos (por ejemplo, una clase tiene un atributo de tipo string pero todos los ítems tienen valores compatibles con el tipo de datos número).

Por lo que se refiere a su definición, hemos introducido un formato de plantilla que ilustramos en la Fig. 7 para el caso de modelado en respuesta a evento externo. Además de cierta información que ya aparecía en el caso de las validaciones y que por razones de espacio no incluimos (v. Fig. 6), consignamos: el evento cuya ocurrencia puede implicar la necesidad de modelado; las condiciones que deben cumplirse al ocurrir el evento; la identificación de los elementos que deben considerarse (sugerencias); la pregunta que debe hacerse al usuario para cada elemento; y la transformación que debe aplicarse sobre cada elemento para el que el usuario haya confirmado la necesidad de modelado.

Nombre: Sugerir Atributos para Clase
Resumen: Sugiere al usuario definir nuevos atributos para la clase asociada a un nuevo ítem
…
Evento que desencadena la comprobación: Creación de un nuevo ítem I:
`Item.allInstances()->includes(I) and I.oclIsNew()`
Condición sobre el evento: El ítem I está clasificado:
`I.class→notEmpty()`
Sugerencias: Nombres de las propiedades del ítem I que no están asociadas a atributos en la clase del ítem I:
`I.property->select(p | I.class.attribute.name->notIncludes(p.name)).name`
Pregunta: Para cada nombre de propiedad sugerido, nameP:
"¿Es cierto que todos los ítems de clase <<I.class.name>> tienen una propiedad <<nameP>>?"
Transformación a aplicar: Para cada nombre de propiedad sugerido que el usuario ha confirmado:
Añadir Atributo desde Propiedad (v. Sección 6)

Fig. 7. Definición de la acción de automodelado Sugerir Atributos para Clase.

8 Conclusiones y Trabajo Futuro

En este artículo se ha propuesto un nuevo paradigma de sistema de información, los MicroIS, que se sitúa en un punto intermedio entre los sistemas de información basados en bases de datos (SIBD) y las hojas de cálculo (HdC). Dicho paradigma, pese a haberse originado en el entorno ofimático, podría aplicarse en entornos más complejos. Por lo que se refiere a los datos, la principal característica del nuevo paradigma es la posibilidad de fluctuar entre diferentes estados de consistencia / estructuración en función de las necesidades de los usuarios. Para ello ha sido necesario estandarizar y documentar el conjunto de transformaciones posibles sobre el esquema de modo que, en todo momento, se mantenga la integridad de la información.

El paradigma que proponemos implica al usuario en el mantenimiento de la integridad de los datos al permitir que éste almacene información que no cumple las

reglas de integridad. Por este motivo consideramos que este paradigma no es aplicable a sistemas donde la información que se maneja sea crítica (debido a la posibilidad de inconsistencias) ni a sistemas en los que el volumen de información sea excesivamente grande para un tratamiento manual e individual de las posibles inconsistencias. Dada nuestra experiencia de más de 10 años en desarrollo de proyectos para organizaciones de todo tipo, las condiciones que hacen de los MicroIS una alternativa atractiva son: Flexibilidad, puesto que al poder variar entre estados de estructuración y consistencia diferentes, se facilita la adaptación a los requisitos cambiantes; bajo coste de desarrollo, ya que es posible desarrollar pequeños sistemas de información sin necesidad de recurrir a especialistas y con una inversión mucho menor que en el caso de los SIBD y un retorno de la inversión mucho más rápido; facilidad de uso, gracias a disponer de una interfaz genérica para todos los MicroIS, que facilita la usabilidad de los mismos.

Algunos de los aspectos del desarrollo de la herramienta no se han cubierto en el presente artículo por cuestiones de espacio, entre ellos la creación de consultas (basada en un lenguaje de expresiones), la creación de vistas (basada en el mismo lenguaje de expresiones que las consultas) o la interfaz de usuario genérica que permite trabajar de manera consistente con diferentes MicroIS al mismo tiempo que establece los mecanismos estándar de personalización (basándose en las vistas)

Nuestro trabajo futuro se dirige a dotar de más potencia las acciones de automodelado, tanto por lo que se refiere a la detección de situaciones como a la sugerencia de alternativas, especialmente por lo que se refiere a la identificación de situaciones no triviales como relaciones de generalización / especialización.

Referencias

1. http://blog.dabbledb.com/2005/03/shouldnt-be-spr.html, último acceso Mayo 2009.
2. http://bnoopy.typepad.com/bnoopy/2005/03/the_long_tail_o.html, último acceso Mayo 2009.
3. Kenneth R. Baker, Lynn Foster-Johnson, Barry Lawson and Stephen G. Powell. “A Survey of MBA Spreadsheet Users”. Disponible on-line en http://mba.tuck.dartmouth.edu/spreadsheet/product_pubs_files.
4. http://dabbledb.com/, último acceso Mayo 2009.
5. Jácome Cunha, João Saraiva, and Joost Visser “From Spreadsheets to Relational Databases and Back”. En *Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, PEPM09, Savannah, Georgia, USA, January 19-20, 2009.
6. Web de Microsoft. “Cómo usar ADO con datos de Excel desde Visual Basic o desde VBA”. Disponible en http://support.microsoft.com/kb/257819, ultimo acceso Mayo 2009.
7. http://andromda.org, último acceso Mayo 2009
8. http://www.openxava.org, último acceso Mayo 2009
9. http://rubyonrails.org, último acceso Mayo 2009
10. Aleix Casagolda. “MicroIS”. Proyecto Fin de Carrera FIB-UPC, 2009.
11. Johnson Ralph, Woolf Bobby, “Type Object Pattern”, *Addison-Wesley Software Pattern Series: Pattern languages of program design 3*, Addison-Wesley Longman Publishing Co., Inc. 1997
12. Scott W. Ambler, Pramodkumar J. Sadalage. *Refactoring Databases: Evolutionary Database Design*. Addison-Wesley Professional 2006

Cache-aware load balancing for question answering

D. Dominguez-Sal[1], M. Surdeanu[2], J. Aguilar-Saborit[3], and J.L. Larriba-Pey[1]

[1] DAMA-UPC, Barcelona, Spain, {ddomings,larri}@ac.upc.edu
[2] Fundació Barcelona Media, Barcelona, Spain, mihai@surdeanu.name
[3] IBM Toronto Lab, Markam, Canada jaguilar@ca.ibm.com

Load balancing algorithms implemented in distributed systems assign the tasks to each node in such a way that all the resources available are used in an even way. Most load balancing algorithms are based on modeling only the CPU, the disk I/O, or both, but none of them is additionally aware of the cache contents in the system. If the cost to process a task is incorrectly estimated, the solutions to rebalance the tasks may be cumbersome, leading either to: (a) aborting a task in the overloaded node, and transferring it to a different node, or (b) migrating a task preemptively from one computer to another. In both cases, the impact on the system is important and it produces processing overhead and additional network communications. Information retrieval systems, which are distributed in clusters, often implement data caches that can reduce significantly the computing time of a task. In this scenario, the load balancing algorithm could overestimate the cost of some tasks, leading to undesired imbalances. The main contribution in this paper is the proposal of two dynamic load balancing algorithms that take into account all the factors that affect the performance of a distributed search engine: the CPU, the I/O and the cache.

The first algorithm proposed, *Probability Cost* (PC), estimates the cost of processing each subtask of a query depending on where the information is located in the distributed cooperative cache of the system, and the current CPU and I/O loads. The cost is more reduced if the node has the data cached locally, larger if it is available through the cooperative cache, and even larger if the data is not available in any of the cluster caches. PC applies this more detailed cost estimation to divide evenly the system workload.

The second algorithm, *Affinity* (AF), additionally takes into account the frequency of accesses to the documents in the past to exploit the data locality for future queries. In addition to the computational cost calculated by PC, AF calculates an *affinity* function (in our implementation, the popular IR metric tf-idf) that scores the similarity between the query and the cache contents of each node. The affinity metric gives preference to nodes that have similar contents to the query, which turns into better local hit rates. Moreover, AF is able to divide unevenly the workload to get a larger benefit of the cache and use more efficiently the disk and the CPU, which also turns into more system throughput.

We compare our proposals to the best previous contributions, Weighted Average Load (WAL) (Surdeanu et al., TPDS 2002). We obtain consistent throughput improvements up to 61% for different query sets, number of nodes, cache sizes, query distributions and search engine configurations.

Abstract of the paper included in the Proceeding of the 17th ACM conference on Information and knowledge management (CIKM), pp. 1271-1280, 2008.

An MDA approach for the development of data warehouses*

Jose-Norberto Mazón and Juan Trujillo

Lucentia Research Group, University of Alicante, Spain
{jnmazon,jtrujillo}@dlsi.ua.es

Different modeling approaches have been proposed to overcome every design pitfall of different data warehouse components. However, they offer partial solutions that deal only with isolated aspects of the data warehouse and do not provide developers with an integrated and standard framework for designing all data warehouse relevant components, such as ETL (Extraction-Transformation-Loading) processes, data sources, data warehouse repository and so on. To overcome this problem, we have described such a framework in this paper. Furthermore, we have focused on the multidimensional modeling of the data warehouse repository. Multidimensional modeling requires specialized design techniques that resemble the traditional database design methods [2]. First, a conceptual design phase is carried out, whose output is an implementation-independent and expressive conceptual multidimensional model for the data warehouse. A logical design phase then aims to obtain a technology-dependent model from the previously defined conceptual multidimensional model. This logical model is the basis for the implementation of the data warehouse.

The definition of a conceptual multidimensional model and how it has been translated to a relational-based logical representation by using a model-driven approach have been described in this work. The advantages of using the model-driven development to design data warehouses are also enumerated. Specifically, we have focus on describing how to build the different MDA artifacts for multidimensional design. To this aim, we have defined an extension of the UML and the CWM. Transformations between models have been also clearly and formally established by using the QVT language.

It is worth noting that other works (such as [3]) have taken advantage of using MDA in database development. However, to the best of our knowledge, this is the first work that proposes the use of MDA to overcome specific multidimensional modeling problems.

References

1. Mazón, J.N., Trujillo, J.: An MDA approach for the development of data warehouses. Decis. Support Syst. **45**(1) (2008) 41–58
2. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo J.: Research in data warehouse modeling and design: dead or alive? DOLAP 2006: 3–10
3. Vela, B., Fernández-Medina, E., Marcos, E., Piattini, M.: Model driven development of secure XML databases. SIGMOD Record 35(3): 22-27 (2006)

* This paper was included in the *Decision Support Systems* journal [1] (IF: 1.873).

Actas JISBD 2009, pp.208, ISBN: 978-84-692-4211-7

An Adaptive Mechanism to Protect Databases against SQL Injection

Cristian I. Pinzón, Juan F. De Paz, Javier Bajo, Juan M. Corchado
Universidad de Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
{cristian_ivanp, fcofds, jbajope, corchado}@usal.es

Abstract. The purpose of this article is to present an adaptive and intelligent mechanism that can handle SQL injection attacks. This proposal focuses on integrating a case-based reasoning (CBR) mechanism with a neural network. The proposed solution thus adapts to changes in attack patterns and provides the ability to detect attacks independently of their evolution. A prototype of the architecture was developed and the results obtained are presented in this study.

Keywords: SQL Injection, database security, case-based reasoning, neural network

1 Introduction

SQL injections are one of the security problems for web solutions that involve unauthorized access to databases [1]. This attack takes place at the database layer when a user request that has been sent through an HTTP request is executed without prior validation.

Various approaches have attempted to deal with the problem of SQL injections [1] [2] [3] [4] [5]. However, the biggest inconvenience of these solutions is their inability to adapt to the rapid changes in attack patterns, which renders them a bit inefficient in the long term. More complex SQL attacks are characterized by the various techniques used for remaining undetected by existing security solutions.

This article presents the SQLCBR classifier. It is a new solution that incorporates a detection strategy that compares attack patterns (signature detection) and a detection pattern that studies the behavior in the technique of the attack (anomaly detection). The former strategy applies an initial filter to detect simple attacks, while the latter focuses on complex attacks that remain unsolved after the first filter. This strategy is based on a CBR reasoning mechanism combined with a Perceptron Multilayer neural network. The CBR system is the key component of the SQLCBR classifier mechanism. The CBR systems are based on the notion that similar problems have similar solutions [6][7]. By combining the CBR mechanism with the neural network, the system we propose is able to learn quickly and adapt to changes in the SQL attack patterns, thus facilitating the task of determining when a user request actually involves a type of SQL injection attack.

Actas JISBD 2009, pp.209-214, ISBN: 978-84-692-4211-7

The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research. Section 3 focuses on the design of the proposed architecture, and section 4 provides a detailed explanation of the classification model. Finally, section 5 presents the results and conclusions obtained by the research.

2 SQL Injection Attack

A SQL injection attack takes place when a hacker changes the semantic or syntactic logic of a SQL text string by inserting SQL keywords or special symbols within the original SQL command that will be executed at the database layer of an application [1]. A SQL injection attack can cause serious damage to an organization, including financial loss, breach of trust with clients, among others.

There have been many proposed solutions for SQL injection attacks, including some Artificial intelligence techniques. One of the approaches is WAVES (Web Application Vulnerability and Error Scaner) [5]. This solution is based on a black-box technique. WAVES is a web crawler that identifies vulnerable points, and then builds attacks that target those points based on a list of patterns and attack techniques. WAVES monitors the response from the application and uses a machine learning technique to improve the attack methodology. WAVES cannot check all the vulnerable points like the traditional penetration testing. The strategy used by the intrusion detection systems has also even been implemented to deal with some SQL injection attacks. Valeur [4] presents an IDS approach that uses a machine learning technique based on a dataset of legal transactions. These are used during the training phase prior to monitoring and classifying malicious accesses. Generally, IDS systems depend on the quality of the training set; a poor training set would result in a large number of false positives and negatives. Skaruz [2] proposes the use of a recurrent neural network (RNN). The detection problem becomes a time serial prediction problem. The main problem with this approach is the large number of false positives and false negatives.

Other strategies based on string analysis techniques and the generation of dynamic models have been proposed as solutions to SQL injection attacks. Halfond and Orso [1] propose AMNESIA (Analysis and Monitoring for Neutralizing SQL Injection Attacks). Kosuga et al. proposes SANIA (Syntactic and Semantic Analysis for Automated Testing against SQL Injection) [3]. With only slight variations of accuracy in the models, these strategies have as drawback their meaningful rate of false positives and negatives

3 Classifier Mechanism Design

The mechanism was strategically designed for classifying user requests into two categories: legal or malicious. The legal requests are forwarded to the database for their execution, while the malicious requests are rejected with an alert activated and sent to security personnel.

The SQLCBR Classifier Mechanism is made up of 3 principle components, as shown in Figure 1.

- The Filter-Traffic component: In charge of capturing traffic directed towards the database Server. JPCAP is the API used to identify and capture any traffic that contains HTTP requests and later facilitating the extraction of the SQL string from the rest of the traffic generated by the request.
- The SQL-Analyzer component: Syntactically analyzes the SQL string passed through the first component. The Cup and JFlex tools are used to extract and format the information required for continuing on to the final component.
- The SQLCBR-Classifier component is the core of the mechanism. The classification mechanism is divided into three subcomponents. The first is a subcomponent that is designed to identify simple attack patterns based on a pattern matching. The application of this initial filter will reduce the number of cases passed on to the CBR engine, thus improving the operation of the solution. The second subcomponent is based on a case based reasoning engine using a neural network, specifically the Multicap Perceptron, for carrying out the classification. Finally, the third subcomponent corresponds to a user interface for interacting with security personnel.

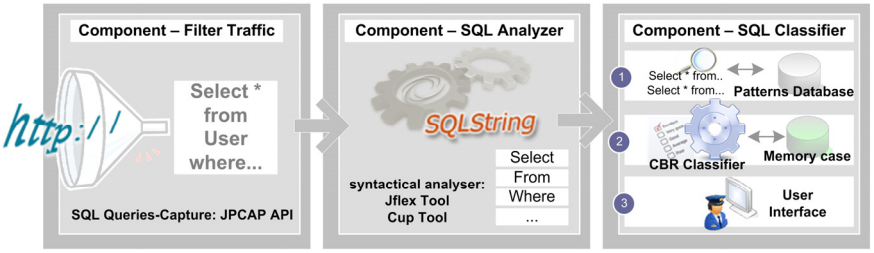


Fig. 1. SQLCBR Classifier Components

4 Mechanism for the Classification of SQL Queries

CBR can be defined as a reasoning model that incorporates problem solving, understanding, and learning, and integrates all of them with memory processes. These tasks are performed using typical situations, called cases, already experienced by a system [6]. Using a CBR focus to solve problems implies the execution of a CBR cycle which is based on 4 known phases: retrieval phase, reuse phase, revise phase, retain phase. A case is defined as a previous experience and is composed of three elements: a description of the problem; a solution; and the final state.

The elements of the SQL query classification are described as follows:

(a) Problem Description that describes the initial information available for generating a plan. Table 1 lists a problem description that consists of a case identification, user session and SQL query elements. (b) Solution that describes the action carried out in order to solve the problem description. Table 1, lists the case

identification and the applied solution. (c) Final State that describes the state achieved after that the solution has been applied.

Table 1. Structure of the problem definition and solution for a case of SQL query classification

Problem description fields			
IdCase	Integer	Numer_And	Integer
Sesion	Session	Numer_Or	Integer
User	String	Number_literals	Integer
IP_Adress	String	Number_LOL	Integer
Query_SQL	Query_SQL	Length_SQL_String	Integer
Affected_table	Integer	Cost_Time_CPU	Float
Affected_field	Integer	Start_Time_Execution	Time
Command_type	Integer	End_Time_Execution	Time
Word_GroupBy	Boolean	Query_Category	Integer
Word_Having	Boolean	Idcase	Integer
Word_OrderBy	Boolean	Classification_Query	Integer

The first step within the classification process consists of submitting the SQL string to a pattern matching in order to identify simple attack patterns. If an association is found between an attack pattern and the SQL request string, the user's request is immediately rejected and an alert is initiated over the security personnel user interface. After this initial filter has been applied to detect simple attack patterns, the number of SQL strings that continue to the SQLCBR classifier is reduced. However, if no anomaly is found among the patterns within the SQL string, the results of the syntactic analysis that was applied on the SQL string are passed on to the SQLCBR Classifier. These results will become the case description of the new problem as shown in Table 1. An example will be presented below.

Let us assume a query with the following syntax for bypassing the authentication mechanism: `Select field1, field2, field3 from table1 where field1='' or 11 = 11 -- 'and field2=''`. The analysis of the SQL string would generate the result presented in Table 2, with the following fields: Affected_table$^{(c1)}$, Affected_field$^{(c2)}$, Command_type$^{(c3)}$, Word_GroupBy$^{(c4)}$, Word_Having$^{(c5)}$, Word_OrderBy$^{(c6)}$, Numer_And$^{(c7)}$, Numer_Or$^{(c8)}$, Number_literals$^{(c9)}$, Length_SQL_String$^{(c10)}$, Number_LOL$^{(c11)}$, Cost_Time_CPU$^{(c12)}$, Query_Category$^{(13)}$. The fields Command_type and Query_Category have been encoding with the following nomenclature Command_Type: 0=select, 1=insert, 2=update, 3=delete; Query_Category: -1=suspicious, 0=illegal, 1=legal, 2= undefined.

Table 2. SQL String transformed through the string analysis

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13
1	3	0	0	0	0	1	1	2	88	1	2,91	2

The first phase of the CBR cycle consists of recovering past experience from the memory of cases, specifically those with a problem description similar to the current SQL query. In order to do this, a cosine similarity-based algorithm is applied, allowing the recovery of those cases which are at least 90% similar to the current SQL query. The recovered cases are used to train the neural networks implemented in the

recovery phase; the neural network with a sigmoidal function is trained with the recovered cases that were an attack or not. A preliminary analysis of correlations is required to determine the number of neurons of the input layer of the neuronal networks. Additionally, it is necessary to normalize the data (i.e., all data must be values in the interval [0.1]).

With the trained neural network we mean to detect complex SQL injection attacks. In the reuse phase the network is trained by a back-propagation algorithm for the set of available training patterns (this particular neural network is named Multilayer Perceptron), using a sigmoidal activation function (which will take values in [0.1], where 0 = malicious and 1 = legal). The number of neurons in the output layer for the Multilayer Perceptrons is 1. It is responsible for deciding whether or not there is an attack. The sigmoidal activation function is given by.

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{1}$$

The review stage is performed by an expert whose final opinion will determine whether the case is stored in the memory of cases and whether the list of well-known patterns has to be updated in the retain phase.

Finally, in the event that the SQLCBR Classifier is unable provide an exact decision, the user's request is rejected. This decision was made after taking into consideration the risk posed by a suspicious query that, if it turned out to be an attack, could be executed within the database.

5 Results and Conclusions

This article has presented a novel solution based on a classification mechanism capable of learning and adapting to changes in the attack patterns of the SQL injections. The proposed solution implements a case-based reasoning engine in conjunction with a neural network. Moreover, an initial filter was implemented based on a pattern matching. This will quickly resolve the simple SQL string requests and allow an improved response time for the solution.

A case study was proposed in order to validate the effectiveness of the SQLCBR classifier prototype. The tests were conducted with a simple web application with database access, MySQL 5.0. The entries were automated by using the SQLMap 0.6.3 tool, with which an initial case base was established for training the SQLCBR-Classifier. In addition the malicious queries to be analyzed by our solution were executed by this tool. Figure 2(a) shows a comparison between the SQLCBR Classifier and other techniques. The empirical results show that the best methods are those that involve the use of a neural network. This method is more accurate than statistical methods for detecting attacks to databases because the behaviour of the hacker is not linear, but dynamic and chaotic. The effectiveness of our solution was demonstrated by the results obtained.

Other data important to note here is the number of training patterns. Figure 2(b) shows the percentage of prediction with regards to the number of patterns in the

training phase. It is clear that a large number the pattern of training improves the percentage of prediction. As we are working with CBR systems, which depend on a larger amount of data stored in the memory of cases for each user, the percentage of success in the prediction increases, as shown in Figure 2(b). CBR systems need to draw from initial information (past experiences) in order to generalize efficient results.

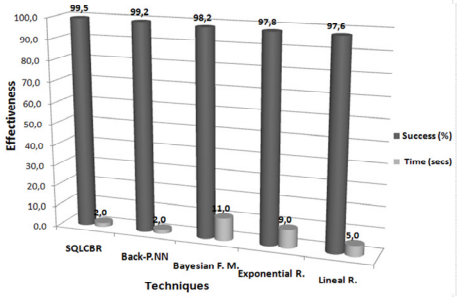


Figure 2(a)

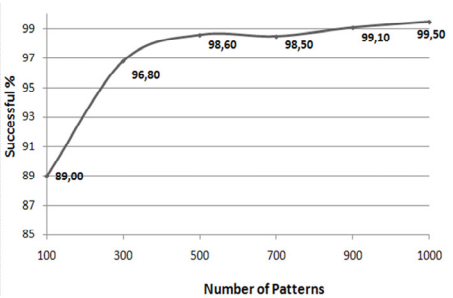


Figure 2(b)

Fig. 2(a). Comparison of the SQLCBR-Classifier with other classification techniques
Fig. 2(b). Sucessful (%) vs Number of patterns

Acknowledgments. This development has been partially supported by the Spanish Ministry of Science project TIN2006-14630-C03-03 and The Professional Excellence Program 2006-2010 IFARHU-SENACYT-Panama

References

1. Halfond, W. and Orso, A.: AMNESIA: Analysis and Monitoring for Neutralizing SQL-injection Attacks. In: 20th IEEE/ACM international Conference on Automated software engineering, pp. 174--183. USA, New York, (2005)
2. Skaruz, J. and Seredynski, F.: Recurrent neural networks towards detection of SQL attacks. In: 21th International Parallel and Distributed Processing Symposium, IEEE International, pp.1--8 (2007)
3. Kosuga Y., Kono K., Hanaoka M., Hishiyama M. and Takahama Y.: Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection. In: 23rd Annual Computer Security Applications Conference, IEEE Computer Society, pp.107--117 (2007)
4. Valeur, F., Mutz, D. and Vigna, G.: A Learning-Based Approach to the Detection of SQL Attacks. In: Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment, pp. 123-140. Vienna, Austria (2005)
5. Huang Y., Huang S., Lin T. and Tsai C.: Web application security assessment by fault injection and behavior monitoring. In: 12th international conference on World Wide Web, ACM, pp. 148--159. New York, USA (2003)
6. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches, AI Commun, vol. 7, pp. 39--59 (1994)
7. Bajo, J., De Paz, Juan F., Corchado, J. M.: Distributed Prediction of Carbon Dioxide Exchange Using CBR-BDI Agents, INFOCOMP, Special Edition, pp.16-25 (2007)

Un Conjunto de *plugins* de *Eclipse* para el Diseño Multidimensional de Almacenes de Datos Dirigido por Modelos

Octavio Glorio, Jesús Pardillo, Paul Hernández, Jose Quinto, Jose-Norberto Mazón, and Juan Trujillo

Grupo de Investigación Lucentia
Dept. de Lenguajes y Sistemas Informáticos, Universidad de Alicante, España
{oglorio,jesuspv,phernandez,jquinto,jnmazon,jtrujillo}@dlsi.ua.es

Los almacenes de datos posibilitan una visión multidimensional de enormes cantidades de datos históricos provenientes de fuentes operacionales, suministrando la información necesaria para el apoyo a los procesos de toma de decisiones de una organización. El paradigma multidimensional estructura la información en hechos y dimensiones. Un hecho contiene medidas interesantes de un proceso de negocio como las ventas o la gestión del inventario (atributos del hecho), mientras que una dimensión representa el contexto de análisis de un hecho (producto, cliente, tiempo, etc.) mediante una serie de atributos organizados jerárquicamente. El modelado multidimensional requiere de técnicas de diseño especializadas que se asemejan a los métodos tradicionales de diseño de bases de datos [1]. En primer lugar se desarrolla una fase de diseño conceptual con el fin de obtener un modelo multidimensional para el almacén de datos independiente de la implementación. Luego, una fase de diseño lógico se encarga de derivar un modelo dependiente de la tecnología a partir del modelo conceptual. Este modelo lógico es la base de la implementación del almacén de datos. Por lo tanto, existen dos tareas fundamentales en el modelado multidimensional:

- **Definición de un modelo multidimensional a nivel conceptual.** Con el fin de que el almacén de datos cubra las necesidades de los usuarios a la vez que se pueda poblar fácilmente con las fuentes de datos, el modelado multidimensional debería tener en cuenta no sólo los requisitos de información del usuario, sino también los datos operacionales existentes.
- **Derivación de una representación lógica.** El proceso de diseño del almacén de datos debería establecer un conjunto de transformaciones formales para obtener automáticamente la implementación final del modelo multidimensional. De esta manera, los diseñadores podrán ahorrar tiempo y esfuerzo en cada proyecto de almacenes de datos desarrollado.

Con el fin de realizar estas tareas de manera satisfactoria, se ha considerado un proceso de desarrollo dirigido por modelos basado en MDA (*Model Driven Architecture*), ya que ofrece mecanismos para gestionar la integración de modelos y para definir transformaciones formales entre ellos que puedan ser ejecutadas automáticamente. La novedad de nuestra propuesta reside en la realización del modelado multidimensional de manera completa, sistemática y estructurada mediante un conjunto de transformaciones formales para obtener automáticamente la implementación final del modelo multidimensional a partir de un modelo conceptual, teniendo en cuenta tanto requisitos de

Actas JISBD 2009, pp.215-218, ISBN: 978-84-692-4211-7

usuario como fuentes de datos operacionales. Tal y como se muestra en la Fig. 1, se define un modelo multidimensional conceptual del almacén de datos (*Platform Independent Model*, PIM) a partir de un modelo de requisitos (*Computation Independent Model*, CIM) que se obtiene de los usuarios del almacén de datos [2]. Este PIM inicial se debe reconciliar con las fuentes de datos [3, 4], obteniendo un nuevo PIM. Además, a partir de este nuevo PIM se pueden derivar varios modelos lógicos (*Platform Specific Models, PSMs)* considerando diferentes plataformas de implementación (relacional, multidimensional, etc.) [5, 6]. Finalmente, el código para la implementación del modelo multidimensional se obtiene a partir de cada PSM.

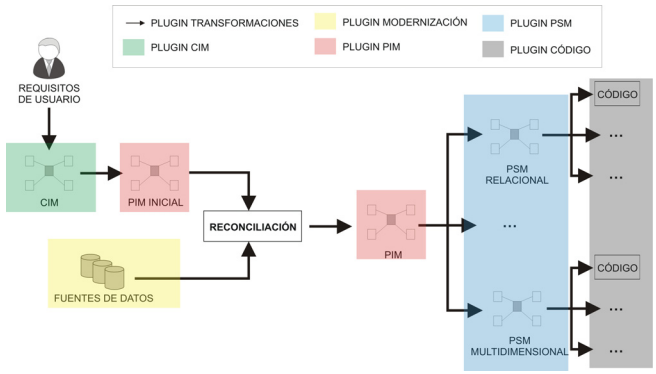


Fig. 1. Propuesta MDA para el modelado multidimensional de almacenes de datos

En concreto, en este artículo se presenta una serie de *plugins* de *Eclipse* para cada una de las partes de nuestra propuesta MDA para el modelado multidimensional de almacenes de datos (ver Fig. 1). A continuación, se resume cada uno de los *plugins* desarrollados.

De manera general, dentro de cada uno de los *plugins* se ha definido diferentes editores textuales y gráficos con el fin de diseñar los diferentes modelos y aplicar las transformaciones necesarias de manera integrada. La Fig. 2 muestra la ejecución de alguno de los *plugins* desarrollados. Cabe destacar que se han creado un par de paletas para poder dibujar los diferentes modelos (Fig. 3(b) y 3(c)). Por otro lado, los diferentes modelos (CIM, PIM y PSM) y el código se guardan en diferentes carpetas cuando se crea un proyecto (Fig. 3(a)). Además, cada una de las transformaciones puede ejecutarse mediante el uso de cada una de las opciones del menú *"Transform"* (Fig. 4).

- ***Plugin* para el CIM.** Este plugin implementa un *profile* UML para definir un CIM basado en *i** en el dominio de los almacenes de datos [2].
- ***Plugin* de modernización.** Este *plugin* permite realizar una conexión a una base de datos *Oracle* y ejecutar las sentencias SQL requeridas para obtener los metadatos del diccionario. Después de obtener los metadatos necesarios, se deriva el modelo correspondiente mediante el uso de *Eclipse* y el metamodelo relacional de CWM

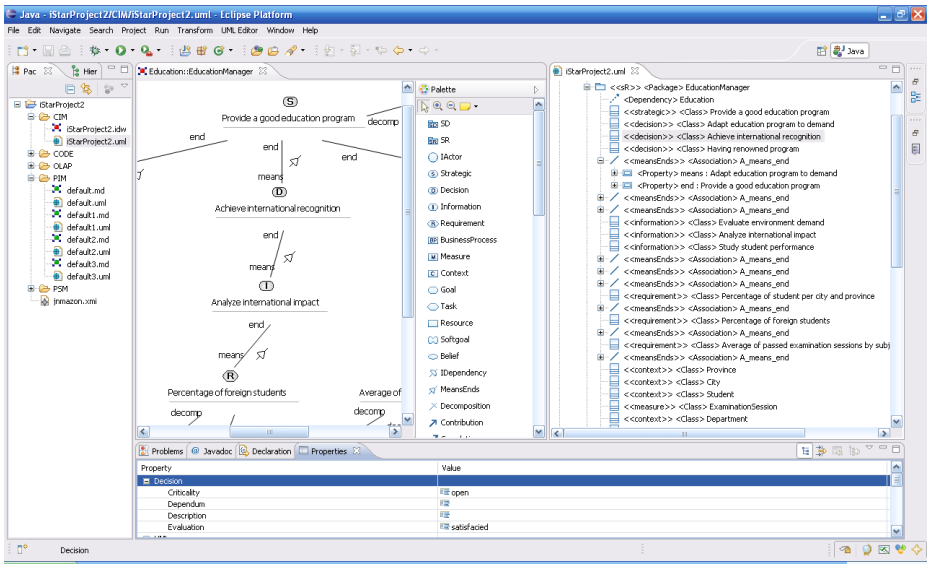
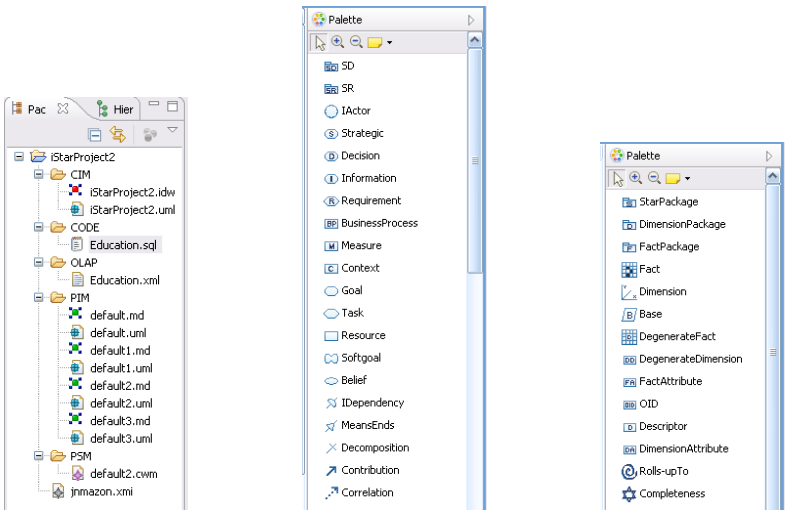


Fig. 2. Captura de pantalla de la utilización del conjunto de *plugins* basados en *Eclipse*



(a) (b) (c)

Fig. 3. Diferentes partes de la herramienta *Eclipse*

(implementado en el *plugin* para los PSM). Cada uno de sus elementos de este modelo se marca con conceptos multidimensionales según unas heurísticas definidas en [4].

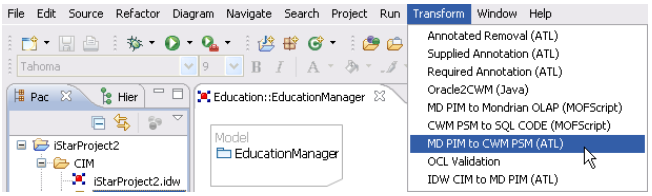


Fig. 4. Menú para la ejecución de transformaciones

- ***Plugin* para el PIM.** Este *plugin* implementa el *profile* de UML para modelado multidimensional [7].
- ***Plugin* para los PSM.** Este *plugin* implementa la capa *Resource* de CWM con el fin de definir modelos lógicos para el almacén de datos, por ejemplo usando tecnología relacional.
- ***Plugin* para las transformaciones.** Se ha optado por implementar las transformaciones definidas en [5, 6] usando el *ATLAS Transformation Language* (ATL). También se ha incluido la reconciliación de fuentes de datos y los requisitos de información según la propuesta definida en [3].
- ***Plugin* para la obtención de código.** Este módulo utiliza el motor de transformaciones *MOFScript* para la generación de código a partir de los modelos.

References

1. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: dead or alive? In: DOLAP. (2006) 3–10
2. Mazón, J.N., Pardillo, J., Trujillo, J.: A model-driven goal-oriented requirement engineering approach for data warehouses. In: ER Workshops. (2007) 255–264
3. Mazón, J.N., Trujillo, J., Lechtenbörger, J.: Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. Data Knowl. Eng. **63**(3) (2007) 725–751
4. Mazón, J.N., Trujillo, J.: A model driven modernization approach for automatically deriving multidimensional models in data warehouses. In: ER. (2007) 56–71
5. Mazón, J.N., Pardillo, J., Trujillo, J.: Applying transformations to model driven data warehouses. In: DaWaK. (2006) 13–22
6. Mazón, J.N., Trujillo, J.: An MDA approach for the development of data warehouses. Decis. Support Syst. **45**(1) (2008) 41–58
7. Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. Data Knowl. Eng. **59**(3) (2006) 725–769

OOH4RIA Tool: Una Herramienta basada en el Desarrollo Dirigido por Modelos para las RIAs[1]

Santiago Meliá, Jose-Javier Martinez, Álvaro Pérez y Jaime Gómez

Universidad de Alicante, IWAD, Campus de San Vicente del Raspeig, Apartado 99 03080 Alicante, Spain
{santi, joseja, aperez, jgomez}@dlsi.ua.es

Resumen. Debido a la necesidad de proporcionar aplicaciones Web con interfaces de usuario cada vez más usables y con mayor funcionalidad, en los últimos años han sugido un nuevo tipo de aplicaciones denominadas RIAs (Rich Internet Applications) que ofrecen interfaces tan interactivos como las aplicaciones de escritorio. Sin embargo, este tipo de aplicaciones son complejas y su desarrollo requiere de un gran esfuerzo de diseño e implementación. Con el objetivo de reducir trabajo y reducir los errores de desarrollo se ha definido una aproximación dirigida por modelos denominada OOH4RIA. Este artículo presenta la herramienta que da soporte a la aproximación denominada OOH4RIA Tool permitiendo representar los modelos y transformaciones para acelerar la obtención de una RIA implementada con el framework GWT.

1 Introducción

Hasta hace pocos años, las aplicaciones Web tradicionales presentaban grandes limitaciones en su interfaz de usuario siendo así más pobres y menos usables que las aplicaciones de escritorio. Sin embargo, recientemente las aplicaciones Web han evolucionado rápidamente gracias a la aparición de nuevas tecnologías que permiten aumentar la calidad del interfaz de usuario de las aplicaciones Web mejorando sus componentes gráficos, reduciendo el trafico con el servidor y aumentando la usabilidad con el cliente final, son las llamadas Rich Internet Applications (RIA).

Sin embargo, las RIAs son aplicaciones más complejas que la Web tradicional y su desarrollo requiere de un diseño e implementación más costosos. Por ello, este nuevo tipo de aplicaciones requiere modificar las metodologías Web, introduciendo nuevos modelos que representen un interfaz más interactivo y mejorar el proceso de desarrollo usando técnicas de automatización que lo aceleren y reduzcan posibles errores.

En este contexto, se define la aproximación OOH4RIA [2] [3] basada en el paradigma del desarrollo del software dirigido por modelos (DSDM) que propone un conjunto de modelos específicos de dominio y un conjunto de transformaciones que permiten obtener la implementación de una RIA. El proceso usa la conocida

[1] Este trabajo ha sido subvencionado por el proyecto ESPIA (TIN2007-67078) del Ministerio de Educación y Ciencia de España

aproximación Web OOH [1] para definir los modelos de dominio y navegación que permiten generar la parte servidora CRUD. Además, ambos modelos son el punto de partida para los modelos de presentación y orquestación que representan la parte cliente RIA. Para poder dar soporte a esta aproximación, se ha implementado una herramienta Web llamada OOH4RIA Tool definida sobre el framework eclipse GMF (Graphical Modeling Framework) que permite representar los modelos de forma gráfica basándonos en una definición formal del metamodelo con el framework EMF (Eclipse Modeling Framework), que permite la utilización de estándares como XMI para el transporte de modelos, MOF para la definición de metamodelos y OCL para introducir restricciones a nivel de metamodelo y validar así la correctitud de los modelos introducidos. Por otro lado, utilizamos el potente framework OpenArchitectureWare como motor de las transformaciones que parte de los modelos EMF y proporciona los lenguajes de transformaciones para obtener la generación de código como resultado del proceso de OOH4RIA.

Para poder comprender correctamente la herramienta, en la sección 2 se presenta una perspectiva general del proceso de desarrollo de OOH4RIA que implementa de forma parcial la herramienta, para finalmente en la sección 3 mostrar las características más importantes de la herramienta.

2 Perspectiva General del Proceso de OOH4RIA

OOH4RIA [2] define un proceso del software dirigido por modelos que pretende cubrir todas las fases de desarrollo para la especificación de una RIA. Se basa en una extensión y mejora de la metodología Web OOH, a la que se le han incorporado los modelos y transformaciones necesarias para obtener una RIA. Para la definición de los modelos se ha utilizado una aproximación DSL (Lenguaje Específico de Dominio) que define sus propios elementos visuales pero basándose en un metamodelo MOF y estableciendo restricciones en OCL.

El proceso comienza con la definición del modelo de dominio OOH que representa las entidades de dominio y las relaciones entre ellas. Este modelo utiliza los mismos elementos y apariencia visual que un diagrama de clases UML, pero define su propio metamodelo para introducir extensiones (como la especificación de la OID, la definición de tipos datos como Date, Double, etc., o tipos de datos complejos como Set, Bag, List, etc.) que permiten mejorar la especificación de la solución implementada. El siguiente modelo de servidor, es el modelo de navegación que permite representar la navegación entre los conceptos de dominio y establece las restricciones de visualización a la capa de interfaz. A partir de estos dos modelos por un lado se definen las transformaciones modelo-a-texto para obtener la parte servidora, y por otro lado son el punto de partida de los modelos de presentación.

Para reducir el esfuerzo en la definición de los modelos de presentación y mantener la coherencia con lo modelos de servidor, se utilizan transformaciones modelo a modelo que permiten obtener esqueletos de dichos modelos de interfaz de usuario.

El primer modelo a nivel de interfaz es el modelo de presentación, que realiza una representación estructural de los componentes visuales que constituyen el interfaz de usuario RIA. Dicho modelo se define como una DSL enfocada en obtener una

apariencia casi idéntica a la interfaz (WYSWYG), consiguiendo que diseñadores o programadores con baja formación en programación puedan utilizarlo. Se representa mediante un conjunto de pantallas denominadas Screenshots que permiten representar la organización de los diferentes widgets en un estado del interfaz. Es importante destacar que estos widgets en ocasiones obtienen o envían información a la parte servidora, esto se realiza estableciendo una relación entre el widget y un elemento del modelo de navegación. Debido a que las RIAs poseen interfaces de usuarios interactivos similares a aplicaciones de escritorio, debemos completar las características estáticas con un modelo que permita especificar la interacción entre los widgets y el resto del sistema. Este modelo es denominado modelo de orquestación y representado como una máquina de estados, donde definiremos un estado para cada screenshot que internamente tendrá subestados que representan los widgets con comportamiento dinámico. Este modelo es explicado en profundidad en [3].

El último paso consiste en definir las transformaciones modelo-a-texto que permitirán obtener la implementación RIA. Por un lado, se define una transformación que genera el código de servidor desde los modelos de dominio y navegación de OOH, mientras que se definirán una transformación que permitirán obtener el código cliente con el Framework GWT. Ambas transformaciones se han definido mediante el lenguaje transformaciones modelo-a-texto Xpand del framework OpenArchitectureWare.

3 Características Principales de OOH4RIA Tool

La herramienta OOH4RIA Tool es una herramienta desarrollada como un plugin del entorno Eclipse, que se fundamenta en el framework GMF (Graphical Modeling Framework) para definir la parte de modelado, y en el framework OpenArchitectureWare para la implementación de las transformaciones en el proceso de desarrollo dirigido por modelos definido por OOH4RIA.

Para la definición del editor gráfico, el framework GMF provee de un entorno de generación y una infraestructura de ejecución que permite un desarrollo rápido de los editores gráficos. Para ello, se basa a su vez, en otros dos conocidos frameworks de eclipse como EMF (Eclipse Modeling Framework) y en GEF (Graphical Editing Framework). Para el desarrollo del editor gráfico de OOH4RIA, se ha definido un metamodelo Ecore que recoge todos los conceptos de los modelos y relaciones entre ellos. A partir de aquí, se ha utilizado el Framework GMF para generar la parte gráfica de cada uno de los modelos, y se ha procedido a la introducción de las restricciones OCL sobre el para evitar posibles combinaciones prohibidas. Este proceso, lejos de ser sencillo ha necesitado modificar el código de GEF para que el modelado y la introducción de las propiedades sobre los elementos visuales fuera la adecuada. La figura 1 muestra una imagen del modelo de presentación sobre la herramienta.

Para el introducción de las transformaciones en la herramienta, se ha utilizado el motor de transformaciones que proporciona el framework OpenArchitectureWare, fundamentalmente este tres lenguajes Xpand para las transformaciones modelo-a-texto, Xtend para las transformaciones modelo-a-modelo y Check para las

comprobaciones en OCL. En OOH4RIA tool hemos integrado dicho motor de transformaciones para que el diseñador no introduzca solo modelos, si no que pueda acceder a las transformaciones y pueda modificarlas en aquellos casos particulares en los que se desee introducir código específico para su solución. Para ello, se genera una copia en las reglas para cada aplicación, y así en el caso de que se corrompan, siempre puedan volver a las reglas que la herramienta proporciona por defecto.

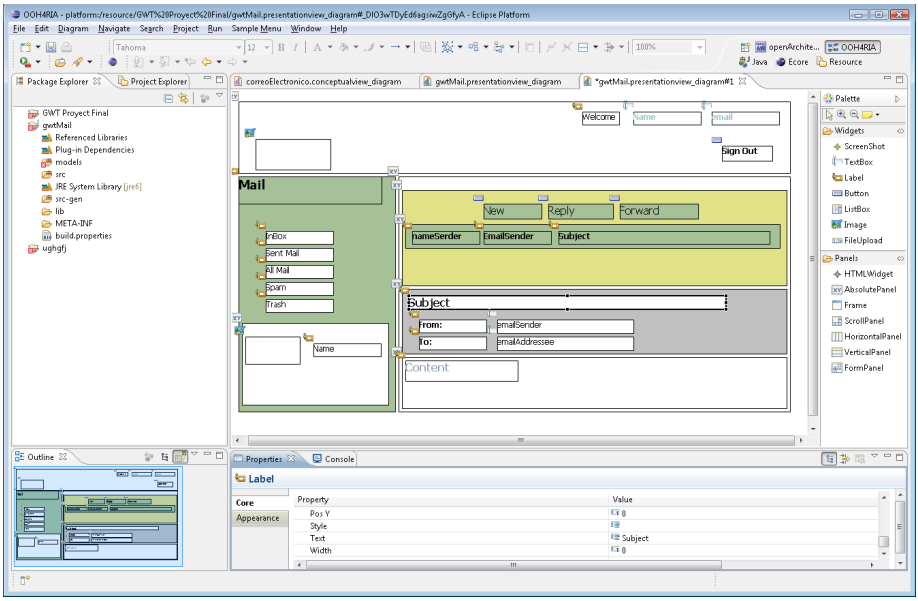


Figura 1. Modelo de presentación en la herramienta OOH4RIA tool

La solución RIA obtenida con OOH4RIA se fundamenta en dos importantes frameworks Java. Por un lado, la RIA obtenida a nivel de interfaz de usuario se implementa mediante GWT (Google Web Toolkit), es un framework AJAX desarrollado por Google que permite crear una RIA escribiendo el código cliente de navegador (HTML, JavaScript) directamente en Java. Por otro lado, en el servidor el código generado se ha centrado en Hibernate, que proporciona un mapeo objeto-relacional permitiendo así definir las reglas de negocio en un modelo de objetos y hacerlos persistentes en una base de datos relacional.

References

1. Gómez J., Cachero C., Pastor O. Conceptual Modeling of Device-Independent Web Applications. IEEE Multimedia, 8(2), 26–39, 2001.
2. Meliá, S., Gómez, J., Pérez, S., Diaz, O. A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA. Eighth International Conference of Web Engineering, Yorktown Heights, USA, 2008.
3. Pérez, S., Díaz, O., Meliá, S. and Gómez, J. Facing Interaction-Rich RIAs: The Orchestration Model Eighth International Conference of Web Engineering, Yorktown Heights, USA, 2008.

Parte VII

Sesión 6. Recuperación de Información, Indexación y BD en Web

Indexación espacial de puntos empleando wavelet trees*

Nieves R. Brisaboa[1], Miguel R. Luaces[1], Gonzalo Navarro[2], and Diego Seco[1]

[1] Laboratorio de Bases de Datos, Universidade da Coruña
Campus de Elviña, 15071, A Coruña, España
{brisaboa,luaces,dseco}@udc.es
[2] Departamento de Ciencias de la Computación, Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
gnavarro@dcc.uchile.cl

Resumen El desarrollo de estructuras de indexación que permitan recuperar objetos espaciales de manera eficiente ha sido un tema de interés en las últimas décadas. La mayoría de estas estructuras han sido diseñadas pensando en las características específicas de la memoria secundaria. Sin embargo, en los últimos años el precio de la memoria principal se ha reducido considerablemente y, por tanto, hoy en día es posible almacenar índices espaciales completos sin necesidad de acceder a disco.

En este trabajo presentamos una estructura para la indexación de puntos diseñada para memoria principal que mantiene un buena relación entre el espacio necesario para almacenar el índice y la eficiencia de las búsquedas. Nuestra estructura se basa en el *wavelet tree*, un árbol diseñado originalmente para indexar los caracteres de un texto, pero que recientemente se ha empleado con éxito para la construcción de auto-índices en áreas tan diferentes como la recuperación de información o la compresión de imágenes.

Key words: Índice espacial, métodos de acceso a puntos, wavelet tree

1. Introducción

La popularidad de los sistemas de información geográfica (SIG) [1] ha aumentado considerablemente en los últimos años. Una de las principales causas de este aumento de popularidad son las mejoras recientes en el hardware que han permitido que el desarrollo de este tipo de sistemas sea abordable por muchas organizaciones. Además, esta tecnología se emplea cada vez más en un mayor número de áreas distintas, como la generación de cartografía, la planificación urbanística, el márketing o la arqueología.

Una de las características más destacables de los sistemas de información geográfica es su capacidad para gestionar enormes cantidades de información. Esto hace que uno de los temas de investigación más importantes en el área sea el diseño de estructuras de indexación que permitan un acceso eficiente a la información. A lo largo de los años, se

* Este trabajo ha sido financiado parcialmente por el Ministerio de Educación y Ciencia (PGE y FEDER) ref. TIN2006-15071-C03-03 y por la Xunta de Galicia ref. 2006/4 y ref. 08SIN009CT.

han propuesto muchas estructuras diferentes para lograr este objetivo. Dichas estructuras se pueden clasificar de manera general en métodos de acceso a puntos (PAMs, del inglés *Point Access Methods*) y métodos de acceso espacial (SAMs, del inglés *spatial access methods*). Los métodos del primer grupo se caracterizan por mejorar el acceso a colecciones de datos formadas por puntos espaciales. En cambio, los métodos del segundo grupo son más generales ya que trabajan con colecciones de todo tipo de objetos geográficos (puntos, líneas, polígonos, etc.).

La mayoría de los métodos propuestos, tanto en una categoría como en la otra, están orientados a su empleo en memoria secundaria. Esto se debe fundamentalmente a razones históricas. Hace años, las memorias eran pequeñas y caras, y esto convertía en inviable el pensar en el diseño de estructuras de indexación espacial que se pudiesen emplear en memoria principal. Sin embargo, en los últimos años, el precio de las memorias se ha reducido considerablemente, aumentando su tamaño también de manera notable. Por tanto, hoy en día es posible diseñar índices espaciales completos que puedan operar en memoria principal. Para ello, a la hora de diseñar nuevas estructuras de indexación no se debe pensar en optimizar tan sólo la eficiencia de las consultas sino también el espacio necesario para el almacenamiento de la estructura.

En este artículo presentamos un nuevo método de acceso a puntos que almacena tanto el índice como la colección de puntos en una estructura compacta. Esta estructura presenta una buena relación entre el espacio necesario para su almacenamiento y la eficiencia de las búsquedas, lo cual la hace adecuada para su empleo en memoria principal.

En los últimos años, la idea de almacenar de forma conjunta los datos y la estructura de indexación que permite un acceso eficiente a los mismos se ha empleado con éxito en varios campos de investigación. Estas estructuras que permiten almacenar los datos y el propio índice de manera conjunta se denominan auto-índices. Por ejemplo, en [2], se presenta una aproximación para indexar documentos empleando *wavelet trees*. Originalmente, el *wavelet tree* [3] se diseñó como un auto-índice organizado en forma de árbol binario para indexar los caracteres de un texto. En nuestro trabajo, analizamos esta estructura y la adaptamos a las características especiales de la información geográfica.

El resto de este artículo está organizado de la siguiente manera. En primer lugar, revisamos el trabajo relacionado en la sección 2. Luego, en la sección 3 describimos nuestra estructura de indexación. A continuación, en la sección 4, presentamos algunos experimentos que hemos realizado para comparar el rendimiento de nuestra estructura frente a otros métodos de acceso a puntos y métodos de acceso espacial en general. Finalmente, cerramos el artículo con las conclusiones y líneas de trabajo futuro en la sección 5.

2. Trabajo relacionado

Durante los últimos años se han propuesto muchos métodos de acceso espacial y muchos métodos de acceso a puntos diferentes. En [4] y [5] se puede encontrar una buena descripción de las estructuras más relevantes en cada categoría. El objetivo de los métodos en ambos grupos es mejorar la eficiencia a la hora de recuperar subconjuntos de datos que se ajustan a una consulta de búsqueda determinada. Una de las consultas de

búsqueda más comunes que deben resolver los métodos de ambos grupos es la consulta de tipo región. Esta consulta define una región en el espacio y obtiene como resultado todos los objetos geográficos indexados que se solapan con dicha región. En la sección 4, comparamos la eficiencia de nuestra estructura frente a la de métodos representativos de las dos categorías de estructuras de indexación espacial a la hora de resolver este tipo de consultas.

El R-tree [6] es uno de los métodos de la categoría SAM más populares y se puede considerar un ejemplo paradigmático. Esta estructura se basa en un árbol balanceado derivado del B-tree que divide el espacio en rectángulos de cobertura mínima (MBRs, del inglés *Minimum Bounding Rectangles*) agrupados jerárquicamente y que pueden solaparse entre ellos o no. El número de nodos hijos de cada nodo interno varía entre un mínimo y un máximo. El árbol se mantiene balanceado dividiendo aquellos nodos que tienen un número de descendientes por encima del umbral de carga máxima y combinando aquellos otros que tienen un número de descendientes por debajo del umbral de carga mínima. Cada nodo hoja tiene asociado un MBR que delimita el área del espacio que cubre ese nodo. Además, los nodos internos también almacenan un MBR que delimita el área que cubren todos sus descendientes. La descomposición del espacio que proporciona el R-tree es adaptativa (es decir, dependiente de la distribución de los objetos geográficos indexados) y puede presentar solapes (es decir, los nodos del árbol pueden representar regiones no disjuntas). Sobre la propuesta original de A. Guttman [6] se han propuesto muchas variantes para mejorar su eficiencia (como el R+-tree o el R*-tree) y para adaptarlas a distintos problemas (como el STR R-tree para colecciones estáticas). En [7] se presenta un resumen de todas estas estructuras y de sus aplicaciones.

Por otra parte, el K-d-tree [8] es uno de los métodos de la categoría PAM que más se ha empleado, debido fundamentalmente a su sencillez y eficiencia. Cuando esta estructura se emplea para indexar colecciones de puntos se denomina más comúnmente Point k-d-tree. En general, el K-d-tree y los métodos derivados de él se basan en árboles de búsqueda binarios que representan una subdivisión recursiva del dominio basada en el valor de un único atributo en cada nivel del árbol. Las numerosas variantes de esta estructura se suelen identificar basándose en cómo particionan el espacio. En nuestros experimentos, empleamos una aproximación estática propuesta en [8] que asume que se conoce de antemano la colección completa de puntos a indexar. En esta variante, las líneas de partición deben pasar obligatoriamente por los puntos del conjunto de datos y los ejes deben alternarse cíclicamente en un orden constante.

Ambas estructuras están optimizadas para minimizar el tiempo necesario para resolver distintos tipos de consultas, fundamentalmente consultas de tipo región. Sin embargo, estas estructuras ignoran completamente el espacio necesario para su almacenamiento. En nuestra opinión, una buena relación entre el espacio necesario para almacenar la estructura y su eficiencia a la hora de realizar búsquedas es más interesante que únicamente la optimización de la eficiencia de las búsquedas. Para realizar esta afirmación nos basamos en que el tiempo de acceso a disco es varios ordenes de magnitud superior al tiempo de acceso a memoria principal. Por tanto, si conseguimos reducir el tamaño de la estructura de indexación, e incluso de los datos, lo suficiente como para que pueda operar en memoria principal con conjuntos reales de datos la eficiencia

de nuestra estructura será mucho mejor que la de cualquier alternativa que tenga que trabajar en disco.

Como ya hemos mencionado, el *wavelet tree* [3] es una estructura compacta diseñada originalmente para indexar los caracteres de un texto. Esta estructura se ha empleado recientemente en otras áreas para almacenar e indexar distintos tipos de información de forma compacta. Por ejemplo, en [2] se emplea para indexar y recuperar documentos, mientras que en [9] se emplea para la indexación de imágenes. La herramienta básica empleada en el *wavelet tree* es la operación *rank* sobre vectores de bits: la consulta $rank(B, i) = rank_1(B, i)$ devuelve el número de bits establecidos a uno en el prefijo $B[1, i]$ de un vector de bits $B[1, n]$. De manera simétrica, $rank_0(B, i) = i - rank_1(B, i)$. La operación dual a $rank_1$ es $select_1(B, j)$, que permite obtener la posición del j-ésimo bit establecido a uno en B. La definición de $select_0$ es análoga. Por ejemplo, dado un bitmap $B = 1000110$, $rank_1(B, 5) = 2$ y $select_0(B, 4) = 7$. Tanto la operación de *rank* como la de *select* se han estudiado mucho en los últimos años y existen implementaciones que permiten su ejecución en tiempo constante y con un consumo de espacio bastante pequeño. Además, la implementación de estas operaciones de forma eficiente continúa siendo un tema de interés hoy en día [10,11].

3. Descripción de la estructura

3.1. Construcción de la estructura

Podemos formalizar el problema que tratamos de abordar de la siguiente manera. Partimos de un conjunto de N puntos, $P = P_1 \dots P_n$, cada uno definido por dos coordenadas (por ejemplo, latitud y longitud) que identifican su posición en el espacio con respecto a un sistema de referencia. Podemos asumir que estos puntos están distribuidos en una matriz de $N \times N$ donde sólo hay un punto en cada fila y columna. Esto no supone una restricción muy importante ya que podemos ordenar los puntos en cada dimensión permitiendo duplicados. En la figura 1 mostramos el proceso de creación de la matriz representativa de los puntos de entrada. Es importante observar que lo único que mantiene la matriz representativa es el orden, no hay distancias ni proporciones. Esto es lo que nos permite crear una matriz que represente cualquier conjunto de puntos aunque algunos tengan coordenadas repetidas.

El *wavelet tree* es una estructura compacta que nos permite almacenar en poco espacio la matriz que acabamos de construir y con unas propiedades interesantes para realizar búsquedas en ella. Considerando una matriz de $N \times N$, podemos construir un *wavelet tree* con $log_2(N)$ niveles y N bits por nivel. Este *wavelet tree* permite almacenar la permutación del orden de los puntos en una dimensión (por ejemplo, longitud) al orden de los puntos en la otra dimensión (por ejemplo, latitud). Sean $X = P_{X1} \dots P_{Xn}$ e $Y = P_{Y1} \dots P_{Yn}$ dichas permutaciones donde los puntos están ordenados por sus longitudes y latitudes respectivamente. Por ejemplo, en la figura 1 podemos nombrar los puntos de izquierda a derecha (es decir, P_i es el i-ésimo punto empezando a contar por la izquierda). Por tanto, la permutación de las longitudes la podemos escribir como $X = P_1 \dots P_n$. En ese caso, la otra permutación la escribiríamos como

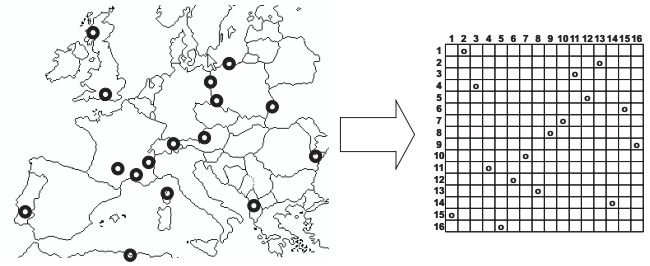


Figura 1. Creación de la matriz representativa de los datos

$Y = P_2P_{13}P_{11}\dots P_1P_5$. Estas permutaciones nos indican, por ejemplo, que el punto P_1 es el primero en el orden de las longitudes y el penúltimo en el orden de las latitudes (es decir, el situado más a la izquierda y el penúltimo más abajo).

La raíz del *wavelet tree* es un bitmap $B = b_1 \dots b_n$ de la misma longitud que el conjunto de puntos (es decir, de N posiciones). Cada posición i del bitmap representa al punto en la i-ésima posición de la primera permutación (es decir, longitud). Siguiendo con el ejemplo, la posición 1 representa el punto $P_{X1} = P_1$, la 2 representa el punto $P_{X2} = P_2$, etc. El valor que se almacena en cada posición puede ser $b_i = 0$ si $P_{Xi} \in P_{Y1} \dots P_{Yn/2}$ o $b_i = 1$ si $P_{Xi} \in P_{Yn/2+1} \dots P_{Yn}$. Es decir, si el punto se encuentra en la primera mitad de la segunda permutación se almacena un 0 y en caso contrario se almacena un 1. La secuencia de puntos marcados con un 1 en el vector se procesan en el hijo derecho del nodo, mientras que aquellos marcados con 0 se procesan en el hijo izquierdo del nodo. En esta estructura, cada nodo indexa la mitad de símbolos que su nodo padre. Este proceso se repite recursivamente en cada nodo hasta que se alcanzan los nodos hoja donde la secuencia de símbolos indexados se corresponde con la permutación en la segunda dimensión (es decir, latitud). En la figura 2 mostramos el *wavelet tree* construido con el ejemplo de la figura 1. Para aportar claridad al ejemplo mostramos para cada elemento del bitmap a qué posición de la segunda permutación se corresponde (estos valores los tachamos para indicar que no se almacenan en la estructura).

3.2. Resolución de consultas

La estructura que acabamos de describir nos permite obtener de forma sencilla en qué posición de la segunda dimensión se encuentra un punto del cual se sabe su posición en la primera dimensión, simplemente descendiendo en el árbol. Para acceder desde una posición determinada de un nodo del árbol al siguiente nivel se emplea la operación $rank$ y el valor almacenado en esa posición. El bit B_i del bitmap de un nodo indica si el punto correspondiente se indexa en el hijo izquierdo ($B_i = 0$) o en el hijo derecho ($B_i = 1$). Además, $rank_{Bi}(B, i)$ nos indica la posición en la que se almacena dicho punto en el nodo hijo. Este proceso se repite hasta que se alcanza un nodo hoja donde la posición señala el orden en la otra dimensión. Siguiendo con el ejemplo, para conocer en qué fila se encuentra el punto de la columna 6 se mira el valor que hay en la posición

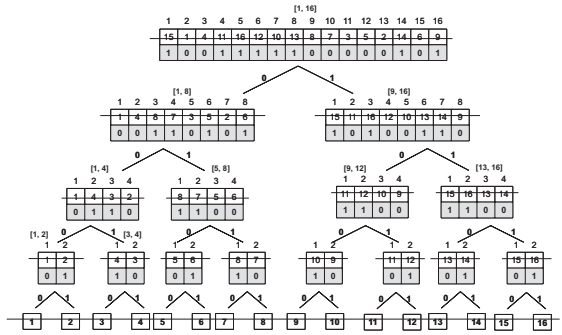


Figura 2. Creación del wavelet tree

6 del nodo raíz y se accede a la posición $rank_1(B, 6) = 4$ del nodo derecho (ya que en la posición 6 hay un 1). Este proceso se repite en todos los niveles del árbol hasta alcanzar el nodo hoja donde se obtiene la posición 12 que, como se puede comprobar en la matriz de la figura 1, es la fila correspondiente a la columna 6.

Del mismo modo, podemos conocer en qué posición de la primera permutación se encuentra un punto del cual se sabe su posición en la otra dimensión simplemente ascendiendo en el árbol. Para acceder al nivel anterior en el árbol desde un nodo se emplea la operación $select$ y el valor que etiqueta la rama del árbol (es decir, el valor que da acceso a ese nodo). Este segundo valor no lo tenemos que almacenar ya que, al ser un árbol binario, es muy fácil de obtener a partir de la posición. En el ejemplo, si queremos conocer en qué columna se encuentra el punto de la fila 13 se mira con qué valor se accedió al nodo que contiene la posición 13 en el último nivel del árbol. Como el último nivel del árbol no se almacena, en dicho nodo se encuentran las posiciones 13 y 14. Además, podemos determinar cuál de las dos posiciones de ese nodo se corresponde con la 13 empleando los valores almacenados en ellas de tal modo que si están en orden en la primera habrá un 0 y en la segunda un 1, y al revés en caso contrario. Por tanto, como sabemos que se accedió a ese nodo con un 0 y que la posición 13 es la primera del nodo (ya que están en orden) se accede a la posición $select_0(B, 1) = 3$ del nodo padre. En el siguiente nivel se calcula $select_1(B, 3) = 6$ (1 por ser un hijo derecho y 3 por ser el valor calculado en el paso anterior) y así hasta el último nivel donde se obtiene la posición $select_1(B, 6) = 8$ que nos dice que el punto se encuentra en la columna 8.

Sin embargo, para poder resolver consultas espaciales de tipo región empleando este índice necesitamos tres estructuras auxiliares: dos vectores con las coordenadas ordenadas en cada dimensión y los identificadores de los puntos ordenados en el mismo orden que uno de esos dos vectores. Esto resulta evidente ya que el *wavelet tree* sólo almacena la relación entre las permutaciones, sin tener para nada en cuenta las relaciones en el mundo real (por ejemplo, las coordenadas de los puntos o las distancias entre ellos). Los vectores con las coordenadas se emplean para traducir la consulta espacial a un rango que indica las columnas y las filas donde se encuentran los puntos que pueden formar parte de la solución de la consulta. Una vez realizada la traducción de la consulta, el rango de columnas (longitudes) indica el rango de posiciones válidas en el

nodo raíz del *wavelet tree*. El descenso en el árbol se realiza tal y como acabamos de explicar para una posición aunque se optimiza considerablemente teniendo en cuenta que los puntos consecutivos se mantienen consecutivos en los nodos hijo. Por tanto, sólo se necesitan dos operaciones de $rank$ (una para la primera posición del rango y otra para la última). Además, el rango de filas (latitudes) se puede emplear para podar el descenso en el árbol. Cada nodo en el *wavelet tree* contiene puntos que cubren un rango determinado de filas. Si ese rango no interseca con el rango de filas que indica la consulta, el algoritmo no tiene que continuar por esa rama.

En la figura 3, mostramos el *wavelet tree* del ejemplo que hemos empleado hasta ahora con los vectores de coordenadas ordenados y los identificadores. En cuanto a estos últimos, sólo necesitamos uno de los dos vectores IDs(X) e IDs(Y) pero la elección de uno u otro condiciona el algoritmo de resolución de consultas y tiene sus ventajas e inconvenientes como veremos a continuación. La figura representa la resolución de un ejemplo de consulta de tipo región $q = \{(27'53, 15'75), (30'71, 19)\}$. Esta consulta se traduce al rango de columnas de interés [6, 10] y al rango de filas de interés [9, 14]. El algoritmo de resolución de consultas comienza con el descenso en el árbol. Como ya hemos mencionado, sólo es necesario calcular los $ranks$ del principio y del final de las posiciones consecutivas. De este modo, el proceso se iniciaría calculando $rank_0(B, 6) = 3$, $rank_0(B, 10) = 4$, $rank_1(B, 6) = 4$ y $rank_1(B, 10) = 6$ (en realidad sólo dos de ellos ya que $rank_0$ y $rank_1$ de la misma posición son dependientes y uno se puede derivar del otro). Por tanto, en el segundo nivel los rangos de interés son el [3, 4] en el hijo izquierdo y el [4, 6] en el derecho. Sin embargo, el hijo izquierdo sabemos que no puede contener soluciones ya que cubre el rango de filas [1, 8] que no interseca con las filas de interés para la consulta [9, 14]. Este proceso se repite en todo el descenso hasta alcanzar los nodos hoja.

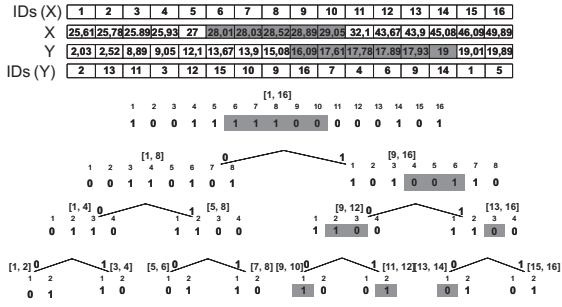


Figura 3. Resolución de consultas empleando el wavelet tree

Una vez que se alcanzan los nodos hoja el algoritmo difiere según el orden en el que se almacenen los identificadores. Si se almacenan en el mismo orden que los nodos hoja (es decir, en el orden de las latitudes), al alcanzar el último nivel del árbol ya sabemos los identificadores que forman parte del resultado de la consulta. Esta versión de la estructura es más sencilla algorítmicamente y como veremos en la sección 4 también

es más eficiente, a pesar de que tiene la desventaja de tener que alcanzar siempre los nodos hoja para obtener el resultado. Por el contrario, si la estructura almacena los identificadores en el mismo orden que el nodo raíz (es decir, en el orden de las longitudes) el algoritmo es más complicado ya que una vez se comprueba que es una latitud válida debe volver a ascender para obtener el identificador. Esta implementación tiene la ventaja de que dicha comprobación se puede cumplir en niveles altos del árbol y, por tanto, se puede detener el descenso y comenzar el ascenso. Sin embargo, los experimentos realizados demuestran que la eficiencia de esta versión es inferior.

4. Experimentos

4.1. Descripción de los experimentos

En esta sección presentamos los experimentos que realizamos para comparar la eficiencia de nuestra estructura frente a otros índices espaciales. Esta comparación tiene dos partes, en primer lugar comparamos los requisitos de espacio de las estructuras y, en segundo lugar, su eficiencia a la hora de resolver consultas espaciales. Como veremos, estos resultados demuestran que nuestra estructura presenta una muy buena relación entre el espacio necesario para almacenarla y la eficiencia a la hora de resolver las consultas.

En los resultados que vamos a presentar a continuación comparamos cuatro estructuras espaciales que trabajan en memoria principal. Las dos primeras se corresponden con las variantes de nuestra estructura de indexación que presentamos en la sección 3. En la primera de ellas, denominada PEW-tree (*efficient point wavelet tree)*, los identificadores de los puntos se almacenan siguiendo la permutación que representan los nodos hoja y, por tanto, sólo es necesario descender en el árbol para obtener los identificadores de los puntos. En la segunda, denominada PCW-tree (*classical point wavelet tree*), los identificadores se almacenan siguiendo la permutación representada en el nodo raíz y, por tanto, una vez que se comprueba en el descenso que un punto pertenece al resultado hay que volver a ascender en el árbol para obtener su identificador. La tercera estructura es el R-tree clásico adaptado para trabajar en memoria principal. Aunque esta estructura se encuentra en la categoría de métodos de acceso espacial en general y no está optimizada para la indexación de puntos, es la más empleada en los sistemas de información geográfica que se desarrollan hoy en día y, por tanto, es importante ver cuánto se podrían beneficiar dichos sistemas empleando nuestra estructura. Por último, la cuarta estructura es un K-d-tree que representa los métodos de acceso a puntos. La variante de K-d-tree que hemos seleccionado es posiblemente la más eficiente ya que está optimizada para situaciones en las que se conoce *a priori* el conjunto de puntos a indexar.

Las colecciones de prueba sobre las que vamos a construir los índices espaciales son colecciones sintéticas con los puntos uniformemente distribuidos en el espacio. Experimentamos con colecciones de puntos con tamaños de 2^{19}, 2^{20}, 2^{23} y 2^{24}.

Para la realización de los experimentos, además de las estructuras de indexación y de las colecciones de puntos, necesitamos un conjunto de ventanas de consulta (es decir, de regiones donde buscar puntos que se encuentren en ellas). Para construirlas, implementamos un algoritmo de generación de ventanas de consulta espaciales de tamaño

parametrizable. Dicho algoritmo está basado en los experimentos diseñados para evaluar el rendimiento del R*-tree tal y como presentaron sus autores en [12]. Las ventanas que generamos están uniformemente distribuidas en el espacio. Además, su proporción se determina de tal forma que el ratio entre la *extensión en x* y la *extensión en y* varía de manera uniforme entre 0,25 y 2,25.

4.2. Comparación de espacio

A la hora de diseñar estructuras de indexación espacial, el espacio necesario para almacenarlas ha sido un factor muy poco valorado ya que la mayoría de estas estructuras estaban pensadas para trabajar en disco. Recientemente, con el crecimiento de las memorias principales algunas de las estructuras clásicas se adaptaron para trabajar en memoria principal pero sin preocuparse demasiado del tamaño que ocupan. Por este motivo, creemos que el desarrollo de estructuras que presenten una buena relación entre el espacio que ocupan y la eficiencia a la hora de resolver consultas es un tópico de especial interés hoy en día.

La estructura que proponemos, en sus dos variantes, necesita almacenar las coordenadas de los N puntos (dos vectores de N elementos que son números en punto flotante), los identificadores (un vector de N elementos que son números enteros) y el *wavelet tree*. El *wavelet tree* es una estructura muy compacta que sólo necesita $N \times log_2(N)$ bits (se necesitan N bits por nivel, un bit por punto y hay $log_2(N)$ niveles). Además, para poder realizar las operaciones de $rank$ y $select$ en tiempo constante se necesitan unas estructuras auxiliares que ocupan un 37,5 % de espacio adicional sobre el tamaño del *wavelet tree*. Es decir, la estructura completa ocupa $20 \times N + (N \times log_2(N) \times 1,375)/8$ bytes.

En cuanto al espacio necesario para la construcción de un R-tree sobre una colección de N puntos, lo podemos estimar suponiendo un factor de ramificación (M) dado. Asumiremos un factor de ramificación de 30 entradas por nodo ya que, en nuestros experimentos, es el que presenta mejores resultados en términos de eficiencia. Además, dado que nuestra estructura es estática, supondremos que los nodos están completamente llenos (en realidad se suele estimar que de media los nodos están al 70 % de su capacidad). En este caso el R-tree consta de $N/M \times (M-1)$ nodos internos y N/M nodos hoja, donde cada nodo tiene un tamaño de $36 \times M$ bytes (los cuatro números en punto flotante que ocupa un MBR y un puntero). Además, hay que sumar el tamaño que ocupa en memoria la tabla de puntos ($20 \times N$ bytes).

Finalmente, un K-d-tree para indexar N puntos tiene una altura $h = \lceil log_2(N) \rceil$ y $2^h - 1 + (N \,\%\, 2^{\lfloor log_2(N) \rfloor})$ nodos, donde cada nodo tiene un tamaño de 16 bytes (un número en punto flotante y dos punteros). Al igual que en el caso del R-tree, hay que tener en cuenta los $20 \times N$ bytes de la tabla de puntos.

En la figura 4 mostramos una gráfica comparativa del espacio requerido por las diferentes estructuras de indexación espacial. Como nuestras dos alternativas ocupan exactamente el mismo espacio mostramos una única curva bajo el nombre de PW-tree. La principal conclusión a la vista de los resultados es que nuestra estructura necesita mucho menos espacio que cualquiera de las alternativas de índices espaciales clásicos por lo que es más adecuada para trabajar en memoria principal.

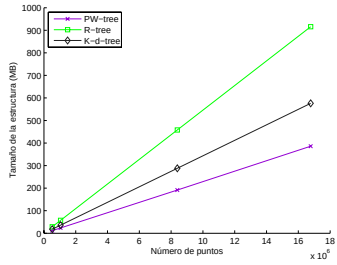


Figura 4. Comparación del espacio requerido por los índices espaciales

4.3. Comparación de tiempo

Para realizar la comparación de tiempos tenemos en cuenta dos parámetros que pueden influir en las pruebas: la selectividad de las consultas y el tamaño de las colecciones. La selectividad de las consultas viene indicado por el tamaño de las ventanas empleadas. En nuestros experimentos creamos ventanas de cuatro tamaños diferentes que representan el 0,01 %, el 0,1 %, el 1 % y el 10 % del área total del espacio donde se encuentran representados los puntos. En la figura 5, mostramos varias gráficas donde se puede observar la influencia de estos factores en el tiempo necesario para resolver las consultas. Los tiempos se midieron en una máquina Intel(R) Pentium(R) 4 3.00GHz con 4GB DDR-400 RAM y sistema operativo Debian GNU/Linux (versión 2.4.27 del kernel).

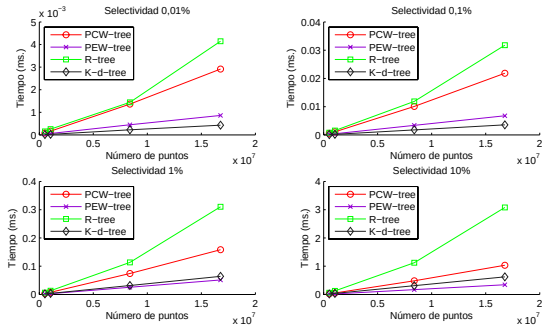


Figura 5. Comparación de tiempos

La conclusión más importante que se puede extraer a la vista de los resultados es que nuestra estructura es competitiva en cuanto a la eficiencia de resolución de consultas, llegando incluso a ganarle al K-d-tree en consultas poco selectivas. El K-d-tree se puede asegurar que es la estructura más eficiente pero, como veíamos en el apartado anterior,

esa eficiencia se basa en un consumo de espacio mucho mayor. Como esperábamos, el R-tree no es tan competitivo ya que es un método mucho más general. Lo incluimos en estos resultados porque es el ejemplo paradigmático de estructura de indexación espacial y, además, se debe tener muy en cuenta ya que sigue siendo la estructura más empleada hoy en día. En cuanto a las dos versiones de nuestra estructura, la PEW-tree (la versión que sólo desciende en la estructura) es más eficiente que la PCW-tree (la versión que necesita ascender en el árbol).

5. Conclusiones y trabajo futuro

En este artículo hemos presentado un nuevo índice espacial en la categoría de los métodos de acceso a puntos. Este índice espacial está basado en el *wavelet tree*, una estructura compacta ampliamente utilizada en áreas como la recuperación de información para la creación de auto-índices. La principal ventaja de esta estructura frente a otros índices espaciales es que presenta una buena relación entre el espacio necesario para almacenarla y la eficiencia de las búsquedas. El poco espacio que necesita le permite operar en memoria principal aun con colecciones realmente grandes de puntos.

Actualmente, estamos trabajando en varias líneas que nos ha abierto esta investigación. En primer lugar, estamos desarrollando una nueva estructura basada en *wavelet trees* para la indexación de cualquier tipo de objeto geográfico empleando sus MBRs. Este índice, más general que el que presentamos en este trabajo, se enmarcaría en la categoría de métodos de acceso espacial. Además, estamos trabajando en la mejora de la estructura de indexación de puntos para permitir la inserción de manera dinámica de nuevos puntos una vez construida la estructura. Aunque consideramos que el ser una estructura estática no es un problema muy grave debido a la sencillez de la estructura y la consecuente rapidez de su construcción. También creemos que es posible mejorar la eficiencia a la hora de resolver consultas y acercarnos un poco más al K-d-tree ya que nuestra estructura se comporta bien en general y tiene un peor caso que se podría mejorar empleando búsquedas binarias. Otra línea de trabajo se centra en la implementación de algoritmos que nos permitan resolver otros tipos de consultas habituales como son las consultas del vecino más próximo (o los k vecinos más próximos) y el *join* espacial. Finalmente, estamos integrando la estructura en sistemas de información geográfica reales para comprobar como mejora la eficiencia de los mismos ya que, aunque se han propuesto muchas estructuras a lo largo de los años, la mayoría de los SIG reales continúan empleando el R-tree para la indexación de cualquier colección de objetos geográficos.

Referencias

1. Worboys, M.F.: GIS: A Computing Perspective. CRC (2004)
2. Brisaboa, N.R., Cillero, Y., Fariña, A., Ladra, S., Pedreira, O.: A new approach for document indexing usingwavelet trees. In: Proc. of DEXA'07. (2007) 69–73
3. Grossi, R., Gupta, A., Vitter, J.: High-order entropy-compressed text indexes. In: Proc. of ACM-SIAM SODA'03. (2003) 841–850
4. Gaede, V., Günther, O.: Multidimensional access methods. ACM Comput. Surv. **30**(2) (1998) 170–231

5. Samet, H.: Multidimensional and Metric Data Structures. M. Kaufmann (2006)
6. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proc. of SIGMOD'84, ACM Press (1984) 47–57
7. Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A.N., Theodoridis, Y.: R-Trees: Theory and Applications. Springer-Verlag New York, Inc. (2005)
8. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9) (1975) 509–517
9. Mäkinen, V., Navarro, G.: On self-indexing images - image compression with added value. In: Proc. of DCC'08, IEEE Computer Society (2008) 422–431
10. Mäkinen, V., Navarro, G.: Rank and select revisited and extended. Theor. Comput. Sci. **387**(3) (2007) 332–347
11. Navarro, G., Mäkinen, V.: Compressed full-text indexes. ACM Comput. Surv. **39**(1) (2007)
12. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. SIGMOD Rec. **19**(2) (1990) 322–331

Desarrollo de un compresor de textos orientado a palabras basado en PPM*

Sandra Álvarez, Ana Cerdeira-Pena, Antonio Fariña, Susana Ladra

Laboratorio de Bases de Datos, Univ. de A Coruña, España.
{salvarezg,acerdeira,fari,sladra}@udc.es

Resumen Reducir el espacio de almacenamiento y el tiempo de transferencia se ha vuelto un aspecto fundamental en las Bases de Datos Textuales. En este trabajo se presenta un nuevo compresor, denominado PPM orientado a palabras (SWPPM), en el que se aplican los modelos estadísticos propios de PPM utilizando como símbolos de entrada las palabras. Presenta varios desafíos técnicos para los que es necesario aplicar nuevas estrategias y diseñar estructuras de datos adaptadas al problema. SWPPM alcanza ratios de compresión del 28 %.

1. Introducción

En los últimos años, las Bases de Datos Textuales han cobrado una gran importancia debido, entre otros factores, a la aparición y rápida expansión de las Bibliotecas Digitales, así como de la Web, que supone la mayor colección de textos existente. Es en este escenario, en donde las técnicas de compresión de textos resultan ser especialmente beneficiosas, ya que no sólo permiten reducir el espacio utilizado en disco, sino también el ancho de banda requerido en la transferencia de los datos y, dependiendo de las técnicas empleadas, también permiten mejorar el tiempo requerido para procesar dicha colección.

Lograr la máxima compresión supone por tanto una gran ventaja en multitud de aspectos, y es en esta línea de investigación donde destaca la técnica PPM (Prediction by Partial Matching)[4], un compresor que obtiene ratios de compresión del orden del 20 %.

En términos generales, las técnicas de compresión se clasifican según sus características. De acuerdo al vocabulario de entrada que utilizan pueden ser orientadas a caracteres o a palabras y, según su alfabeto de salida, pueden ser orientadas a bit o a byte. Además, los compresores pueden considerar un número fijo de símbolos de entrada para codificar, como es el caso general de los compresores estadísticos, o un número variable, típico de las técnicas basadas en diccionario [11]. Por último, los códigos resultantes de comprimir cada símbolo de entrada pueden tener longitud fija o variable.

Los compresores estadísticos utilizan un modelo del texto (básicamente símbolos de entrada y sus frecuencias) para asignar códigos de menor longitud a aquellos símbolos

* Parcialmente financiado por: el "Ministerio de Educación y Ciencia"(PGE y FEDER) ref. TIN2006-15071-C03-03; "Xunta de Galicia", ref. 2006/4; y el "Ministerio de Ciencia e Innovaciónref. AP2007-02484 (Programa FPU) para Ana Cerdeira-Pena

que posean una mayor frecuencia. Estas frecuencias pueden venir prefijadas independientemente del texto que se comprima (*técnicas estáticas*) o depender del texto que se va a comprimir (técnicas semiestáticas o dinámicas) [1]. Las técnicas semiestáticas realizan una pasada previa sobre el texto para recopilar los diferentes símbolos existentes y sus frecuencias y utilizarlos para codificarlos adecuadamente en una segunda pasada. Por su parte, las técnicas dinámicas realizan una única pasada. En ella se va actualizando el modelo del texto que es utilizado para codificar los símbolos a medida que se van procesando [1]. El descompresor partirá del modelo previo en las técnicas estáticas y semiestáticas, mientras que en las técnicas dinámicas lo reconstruye del mismo modo que el compresor, a medida que descomprime el texto.

En función del modelo estadístico utilizado, hablamos de *modelos de orden 0*, cuando los símbolos de entrada se consideran de modo independiente (por ejemplo, Huffman [5]); y *modelos de orden n*, cuando la probabilidad de ocurrencia de cada símbolo viene determinada por el contexto de los n símbolos que lo preceden.

A lo largo de la historia han surgido numerosos compresores estadísticos orientados a carácter. La técnica de Huffman [5] es quizás el representante más conocido de esta familia de compresores, aunque sus ratios de compresión son pobres (en torno al 60 %), lejos del 20 % del PPM.

La creación de compresores orientados a palabras pueden mejorar las tasas de compresión [6]. Adaptaciones de la técnica clásica de Huffman, como la descrita en [9], las conocidas como Plain Huffman y Tagged Huffman [10] o técnicas con similares características como el End-Tagged Dense Code [2,3], mejoran los ratios de compresión obtenidos con Huffman clásico hasta un 30 %. Estas técnicas (orientadas a palabras), utilizan modelos de frecuencia de orden 0, es decir, no tienen en cuenta las palabras precedentes para calcular el modelo de frecuencias.

Es conocido que las técnica PPM, que utilizan modelos de órdenes elevados, obtienen muy buenos ratios de compresión. Además, se han visto, por ejemplo con el caso del Huffman clásico, que tomar una técnica orientada a caracteres para producir una orientada a palabras es efectivo, mejorando el ratio de compresión. De aquí surge nuestra propuesta de realizar un compresor semiestático orientado a palabras basado en PPM (SWPPM), que utilice contextos, formados por las palabras precedentes, para calcular la probabilidad de cada símbolo.

2. Conceptos básicos

Prediction by Partial Matching PPM es un compresor estadístico cuya idea fundamental consiste en utilizar los contextos de los últimos símbolos que han sido procesados para predecir cuál es el símbolo que los sucede. Para ello se mantienen modelos de diferentes órdenes, que se traducen en contextos de tamaño variable, utilizando para codificar el símbolo aquel orden de mayor magnitud del que dispongamos información [8]. Naturalmente, uno de los aspectos más críticos de la implementación de esta técnica es cómo almacenar los contextos y los símbolos que lo siguen (y con qué frecuencia).

El proceso básico de compresión consiste en utilizar el contexto de mayor orden para calcular la probabilidad del símbolo que se va a codificar, emitiendo símbolos de

escape en todos aquellos órdenes en los que no se disponga de información sobre el símbolo.

PPM es un compresor dinámico orientado a caracteres, por lo que la información contenida en los modelos de diferentes órdenes se va modificando durante la compresión, añadiendo nuevos contextos con sus símbolos y probabilidades cuando sea necesario, y actualizando las probabilidades de los ya existentes. Esto produce un tiempo de adaptación inicial en los primeros pasos de la compresión, que provoca que los símbolos se codifiquen de forma poco eficiente mientras no se disponga de suficiente información sobre los contextos.

Una vez se disponga de la probabilidad del símbolo actual en el contexto que se pretende codificar, se le proporciona dicha información a un módulo de codificación aritmética, que es el que proporciona el texto comprimido final. Cuando no se dispone de la información necesaria para un símbolo en un contexto dado, es necesario emitir un símbolo de escape. que es necesario emitir cuando no se dispone de la información Dicho símbolo ha de ser codificado como un símbolo más en dicho contexto, por lo que debe tener asignada una determinada probabilidad en cada contexto.

Codificación aritmética Esta es una técnica estadística que, partiendo de un conjunto de símbolos de entrada produce como salida un número entre 0 y 1 que lo identifica unívocamente [7].

La idea principal consiste en mantener un intervalo, que representará en todo momento a los símbolos ya procesados. El intervalo se divide en subintervalos de tamaño proporcional a las probabilidades de los símbolos, convirtiéndose en el intervalo actual aquel subintervalo que se corresponda con el símbolo a codificar.

Esta codificación resulta ser muy adecuada para PPM, puesto que permite definir en cada paso qué probabilidad se le asigna a cada símbolo y cuál es su codificación resultante. Para ello, PPM debe proporcionar en cada paso la frecuencia del símbolo en el contexto actual, la frecuencia total del contexto y la frecuencia acumulada inferior del símbolo, por lo que es necesario determinar un orden de los símbolos dentro del contexto.

3. PPM orientado a palabra (SWPPM)

El factor determinante en este desarrollo ha sido el diseño de una estructura de datos que sea eficiente y compacta, esto es, eficiente para recuperar la información de los contextos y compacta para ser almacenada en disco como parte del texto comprimido final, sin comprometer el rendimiento del compresor.

PPM utiliza contextos de orden $0, ..., n$, por lo que necesita tener recopilada toda esta información durante la compresión. En un compresor orientado a caracteres, el número de contextos diferentes está relativamente acotado, debido a que el número de símbolos de entrada está limitado a 256 caracteres. Sin embargo, utilizar palabras aumenta este número en varios órdenes de magnitud.

La familia de compresores PPM sigue una aproximación dinámica. En un compresor orientado a palabras, el tiempo de adaptación inicial sería demasiado elevado,

debido al gran número de contextos diferentes que existen. Es por esto que se ha implementado un compresor semiestático. En una primera pasada sobre el texto se recopila toda la información sobre los contextos y sus probabilidades, utilizando toda esta información en una segunda pasada para codificar el texto.

3.1. Estructura del compresor

El proceso de compresión se divide en dos fases principales:

Cálculo de probabilidades Se realiza una primera pasada sobre el texto, calculando las frecuencias de todas las palabras. Como se trata de un modelo orden-n, las probabilidades estarán condicionadas por las $0, ..., n$ palabras precedentes.

Compresión En una segunda pasada sobre el texto, se utilizan los contextos y símbolos extraídos para comprimir el texto, utilizando siempre, al codificar un símbolo, el contexto de mayor orden del que se disponga información. Además, el texto comprimido final debe incluir los contextos, símbolos y frecuencias que se han utilizado, para poder realizar posteriormente la descompresión.

En una primera fase del desarrollo se decidió probar los resultados de compresión obtenidos almacenando todos los contextos encontrados, pero no se obtuvieron resultados competitivos. Por lo tanto, se decidió aplicar un *filtro de contextos* tras el cálculo de los modelos completos, eliminando aquellas cadenas de texto con una baja frecuencia, ya que previsiblemente no se utilizarán un gran número de veces. Así, el coste de almacenarlos en el texto comprimido final puede no ser compensado con el beneficio en la compresión que proporcionan. Es necesario determinar una frecuencia límite que marque qué elementos se eliminarán, denominada *umbral*. Su valor se determinará experimentalmente (ver Sección 5).

Tras realizar el filtrado, el proceso para comprimir consiste en analizar el contexto de longitud n de la palabra a codificar. Si no se encuentra información sobre este contexto, se desciende al orden inmediatamente inferior (consultando las $n - 1$ palabras anteriores), y se repite el proceso. Si por el contrario se encuentra el contexto pero no se dispone información sobre el símbolo actual, se emite un símbolo de escape y se desciende del mismo modo al orden inferior para volver a realizar la comprobación con un contexto de la longitud inmediatamente inferior. Por último, si el símbolo aparece en este contexto, se utiliza el codificador aritmético proporcionándole la probabilidad de la palabra en su contexto.

Nótese que a pesar de tratarse de un compresor semiestático, sigue siendo necesaria la emisión de símbolos de escape, ya que al haber eliminado los símbolos que aparecen un número reducido de veces en el contexto, puede ser necesario descender de orden, e indicarle al descompresor este hecho. Por tanto, es necesario que el símbolo de escape disponga de una frecuencia asignada en ese contexto.

Una vez se ha comprimido el texto, es necesario representar los contextos y sus frecuencias de una forma compacta, ya que es necesario que formen parte del texto comprimido final para permitir la posterior descompresión.

3.2. Estructura del descompresor

El proceso de recuperación del texto original a partir del texto comprimido se realiza de forma simétrica al proceso de compresión. El cálculo de probabilidades se simplifica, puesto que sólo se han de recuperar los contextos que el compresor ha producido como salida en la compresión. Una vez se dispone de toda esta información, descomprimir el texto supone básicamente realizar un proceso análogo al de la compresión, descodificando siempre en el contexto de mayor orden posible, y si el símbolo descodificado se trata de un símbolo de escape, descendiendo de orden y repitiendo el mismo proceso.

4. Diseño de estructuras para SWPPM

4.1. Almacenamiento de contextos

Toda la información sobre los modelos de orden $0, ..., n$ debe ser almacenada en una estructura que permita un rápido acceso durante la fase de cálculo de contextos y de compresión. En el almacenamiento de contextos, se precisa conocer si una secuencia de palabras (contexto y la palabra actual) ya existe en la estructura, para modificar su frecuencia o, en caso contrario, para insertarla con frecuencia 1.

Al igual que PPM orientado a caracteres, SWPPM también utiliza codificación aritmética. Así, para calcular el subintervalo que le corresponde al símbolo actual y poder codificarlo, se necesita obtener la *frecuencia absoluta del contexto*, la *frecuencia del elemento en ese contexto* y la *frecuencia acumulada inferior* a ese símbolo, esto es, de los otros símbolos anteriores en el contexto actual. Para ello, es preciso determinar un orden para los símbolos del mismo contexto, y tener acceso a ellos para poder sumar sus frecuencias y así calcular su frecuencia acumulada inferior. Estos tres valores son los que requiere el codificador aritmético para procesar el símbolo.

Se ha optado por almacenar los contextos en una *tabla hash*, donde cada elemento contiene la información de una cadena de texto, que está formada por un contexto y un símbolo que le sigue, con su frecuencia. Un contexto de k palabras se almacena recursivamente como una referencia al contexto de $k - 1$ palabras y el símbolo correspondiente. El símbolo es una referencia a la palabra en el *vocabulario* almacenado de forma independiente.

Para el cálculo de la frecuencia inferior es necesario disponer de información que relacione entre sí los elementos que comparten el contexto. Para ello, cada elemento contiene otras dos referencias a otros elementos de la tabla: *SiguienteContexto* y *PrimerDescendiente*. De esta forma, se implementa un árbol de contextos sobre la tabla hash. La función hash se calcula a partir de las palabras que forman el contexto y la palabra que sigue a dicho contexto.

4.2. Filtro de contextos

Se descartan los contextos que no superen un determinado umbral de fre- cuencia, de forma que no serán utilizados para la compresión. Al eliminar un contexto, también se tienen que eliminar los contextos que desciendan de él en el árbol de contextos. Nótese que esto es consistente, pues si un contexto tiene una frecuencia menor al umbral, necesariamente los contextos que desciendan de él serán poco frecuentes.

Un aspecto crítico en la eliminación de contextos consiste en mantener el árbol de contextos correctamente enlazado. El proceso de eliminación de contextos se hará de la siguiente forma:

- **Paso 1:** Se asigna el valor 0 a la frecuencia de aquellos elementos que no superen el umbral, siempre que no sean de orden 0 (que representa la frecuencia total de cada palabra independientemente de su contexto), ya que se precisa un orden en el que los elementos se puedan codificar con total seguridad (como se verá en el proceso de compresión).
- **Paso 2:** Se restauran los punteros *PrimerDescendiente*. Si apuntan a una fila cuya frecuencia sea 0, se recorre la lista de ese contexto mediante las referencias de *SiguienteContexto* hasta que se encuentre una fila cuya frecuencia sea distinta de 0, asignando la referencia de esa fila a *PrimerDescendiente*.
- **Paso 3:** Se restauran los punteros *SiguienteContexto* del mismo modo que se realizó con *PrimerDescendiente*.

Para permitir posteriormente un almacenamiento compacto de estas estructuras, resulta conveniente que la lista de elementos que comparten contexto (representada por punteros *PrimerDescendiente* y *SiguienteContexto*) esté ordenada alfabéticamente según las palabras que representan. Esta operación modifica solamente los valores de los campos *PrimerDescendiente*, *SiguienteContexto* y la referencia a la palabra, puesto que también se reordena el vocabulario. El proceso consta de varias fases:

Ordenación del vocabulario: en esta fase se reordena el vocabulario alfabéticamente y se proporciona un vector de *equivalencias* para poder realizar posteriormente la actualización de las referencias a la palabra.

Ordenación de las listas: una vez ordenado el vocabulario, el proceso de ordenación de las listas se vuelve sencillo. Para cada elemento, se sigue su lista accediendo a su campo *PrimerDescendiente* y posteriormente utilizando los enlaces con *SiguienteContexto*, almacenando los valores de palabra que se vayan procesando. Dado que el vocabulario está almacenado ahora alfabéticamente, ordenar los valores de palabra consiste en ordenar numéricamente ese vector. Una vez realizada esta ordenación, se recorre de nuevo la lista asignando por orden los valores de palabra encontrados a las referencias *SiguienteContexto*.

Al comenzar a realizar el procesamiento del texto, no se conoce el número de elementos que va a contener la *tabla de contextos*. Una estrategia simple consiste en crear una tabla que ocupe la mitad de la memoria disponible. Una vez calculados los contextos, y descartados los que aparecen pocas veces, se puede reajustar a un tamaño apropiado (por ejemplo a dos veces la cantidad de contextos).

4.3. Compresión

En este proceso, con toda la información disponible de la primera pasada sobre el texto, se realiza una segunda pasada en la que se codifica cada símbolo mediante el contexto de mayor orden posible, emitiendo un símbolo de escape si no se dispone información de los órdenes superiores.

En el proceso de codificación se utiliza el contexto de mayor longitud del que se disponga de información. En el caso de que no aparezca este contexto seguido del símbolo procesado, se enviará un símbolo de escape. Esta emisión puede ser muy elevada si se eliminan muchos elementos de la tabla de contextos (umbral alto). A la hora de la codificación, el codificador aritmético manipula este símbolo del mismo modo que las palabras. La frecuencia asignada al símbolo de escape influye en la frecuencia total del contexto, con lo que asignar una frecuencia óptima a dicho símbolo repercutirá en la eficacia del compresor.

Para ello, hemos estudiado dos alternativas. La primera consiste en asignar al símbolo de escape una frecuencia dada como porcentaje de la suma de las frecuencias de los símbolos existentes para ese contexto (*porcentaje del intervalo*). La segunda opción asigna al símbolo de escape la suma de todas las frecuencias de símbolos que hayan sido eliminados en un determinado contexto, ya que ésta será una medida bastante aproximada de las veces que el símbolo de escape se emitirá (*frecuencias eliminadas*). Experimentos preliminares muestran que con esta última aproximación se obtienen unos mejores resultados de compresión, y será la que utilicemos en la Sección 5.

Para calcular la frecuencia acumulada inferior de cada símbolo, es necesario recorrer la lista de símbolos de un contexto, formada por la referencia *PrimerDescendiente* del contexto y *SiguienteContexto* de los elementos que lo siguen. Este proceso ralentiza la compresión, por lo que se realizó una mejora consistente en calcular previamente las frecuencias acumuladas inferiores de los símbolos, así como la de los símbolos de escape. De este modo, recorriendo cada lista una sola vez se almacenan todos los valores necesarios, permitiendo en el proceso de compresión realizar la codificación accediendo directamente al elemento necesario y al elemento de su contexto.

4.4. Compresión de las estructuras auxiliares

El compresor y el descompresor precisan disponer del conjunto de elementos que serán utilizados para realizar la compresión y la descompresión. Así, partiendo de la tabla de contextos creada, es necesario transformar esa misma información de una forma que ocupe el menor espacio posible en disco.

El proceso consiste en dividir dicha tabla en vectores siguiendo una aproximación diferencial (es decir, se almacena la diferencia con respecto al valor inmediatamente anterior) que da lugar a valores bajos y repetitivos, y a continuación aplica compresión Huffman, sobre las diferencias. Para ello, se precisa tener ordenadas alfabéticamente las listas de elementos que comparten el contexto.

Estructura intermedia En primer lugar, se precisa transformar la tabla a una estructura intermedia que facilite la conversión a los vectores finales. Dicha estructura contendrá los elementos (contextos y las palabras que los siguen) ordenados según el orden del contexto y posteriormente por orden alfabético. Esta estructura está formada por varios campos:

- *Índice*: indica en qué fila de la estructura original se encuentra el elemento.
- *Contexto*: indica en qué fila de esta estructura intermedia está el contexto del elemento. Para los elementos de orden 0, este elemento apunta al vocabulario. Para los demás, este valor hace referencia a un elemento de orden inmediatamente inferior.

- *Palabra*: indica en qué fila de esta estructura está la palabra del elemento. En el caso de los elementos de orden 0 este valor es nulo, y en el resto de los casos, hace referencia a un elemento de orden 0.

Vectores finales Con el apoyo de esta estructura, se calculan los vectores que posteriormente serán comprimidos con Huffman. Tal y como se ha mencionado previamente, se utiliza una aproximación incremental. Se debe calcular:

- *LongitCont*: longitud del vector *Contextos*.
- *TotalSeg*: la posición i indica el número de elementos en el vector $Segundos$ que hay de orden $0, ..., i-1$.
- *Contextos*: para cada elemento de la estructura intermedia de orden superior a 0, indica la fila de dicha estructura en la que se encuentra el contexto de ese elemento. Las posiciones son relativas, es decir, si el contexto es de orden n, el número indica el incremento de posición desde el primer elemento de orden n que hay en la fila intermedia. Esta estructura sólo almacena los contextos diferentes.
- *Numveces*: número de descendiente del contexto que ocupa esa misma posición en *Contextos*.
- *Segundos*: indica las palabras que suceden al contexto definido por *Contextos* y *NumVeces*. Para cada contexto, el primer elemento en este vector se corresponde con la posición de la palabra en la estructura intermedia en el orden 0. Las siguientes palabras con el mismo contexto se codifican con la aproximación incremental. Esto produce números relativamente y bajos y con un mayor número de repeticiones, debido a que las cadenas de texto están en orden alfabético, y la mayoría de los elementos de este vector son la distancia entre elementos próximos de orden 0.
- *Frecuencias*: frecuencia de todos los elementos de la estructura. La frecuencia se almacena para los elementos de todos los órdenes, incluido el orden 0, por lo que la longitud de este vector es $TotalSeq[n+1]$.

En la Figura 1 se muestra el proceso de compactación de las estructuras auxiliares que posteriormente serán comprimidas usando Huffman.

	IND.	CONT.	PALAB.	
A	8	0	Ø	*ORD. 0*
B	17	2	Ø	
C	7	4	Ø	
D	47	6	Ø	
A A	5	0	0	*ORD. 1*
A C	25	0	4	
B A	27	2	0	
B B	15	2	2	
B C	56	2	4	
AA B	2	4	2	*ORD. 2*
AA D	9	4	6	
BB A	13	7	0	
BB D	21	7	6	
BC D	58	8	6	

LON PRIM:	5											
TOTALSEG	0	4	9	14								
	Ord 0	Ord 1	Ord 2	Ord 3								
CONTEXTOS	0	1	0	3	1							
	Ord 1		Ord 2									
NUMVECES	2	3	2	2	1							
	Ord 1		Ord 2									
SEGUNDOS	0	2	0	1	1	1	2	0	3	4		
	Ord 1					Ord 2						
FRECUENCIAS	30	20	20	25	12	15	7	16	8	6	5	5	6	5
	Ord 0					Ord 1					Ord 2	

Figura 1. Ejemplo de compactación de las estructuras auxiliares

En el conjunto de vectores finales de orden 0 sólo se almacenan las frecuencias. Esto es posible porque en el orden 0 se almacenan todas las palabras por orden alfabético,

y esta ordenación ya se mantiene en el vocabulario, por lo que no es necesario volcar en disco ninguna información más sobre los elementos de este orden para poder reconstruirlos posteriormente.

Tras obtener los vectores finales que han de ser volcados a disco, un último paso previo al volcado consiste en comprimir los vectores de mayor longitud (*Contextos, NumVeces, Segundos* y *Frecuencias*) aplicando Huffman a cada uno de los vectores independientemente. Así, para cada vector se calcula la frecuencia de cada valor, y con ello se crea el árbol Huffman correspondiente. Con este árbol se codifica el vector y se vuelca a disco, ya comprimido, incluyendo también el árbol de Huffman, necesario para la posterior descompresión.

4.5. Descompresión

El proceso de descompresión se realiza de forma muy similar a la compresión. Una vez realizada la recuperación de contextos, se dispone de la tabla de contextos y el código para descomprimir. El proceso es el siguiente: tras indicarle al contexto la frecuencia total del mismo, el decodificador aritmético devuelve un número. El intervalo al que pertenezca dicho número indica el símbolo descodificado, y en caso de que se corresponda con un símbolo de escape, se desciende de nivel, indicándole la frecuencia del contexto de orden inferior y comprobando de nuevo a qué subintervalo pertenece el número devuelto por el descodificador.

5. Resultados experimentales

Los experimentos se ejecutaron en una máquina Intel®Pentium®-IV@ 3.00 GHz (16Kb L1 + 1024Kb L2 cache), con 4 GB dual-channel DDR-400Mhz RAM, sobre la que corría Debian GNU/Linux (kernel version 2.4.27). El compilador usado fue *gcc* version 3.3.5. Las pruebas, que miden el ratio de compresión, fueron realizadas sobre un conjunto de textos de la "Text Retrieval Conference" [1] (TREC); de TREC-2 se han utilizado "AP Newswire 1988"y "Ziff Data 1989-1990", y de TREC-4 "Congressional Record 1993", "Financial Times 1991, 1992, 1993 y 1994". A su vez *FTALL* agrega los corpus provenientes del Financial Times. Además, se utilizó el texto *English50* [2], resultado de los primeros 50 MegaBytes del texto *English*, consistente en una concatenación de textos en inglés del proyecto Gutemberg [3]

En un primer lugar, se analizó la influencia del orden máximo utilizado en el compresor. La Gráfica izquierda de la Figura 2 muestra los resultados obtenidos para el texto *English50*. Se muestra el orden máximo frente al ratio de compresión. Cada curva representa los ratios obtenidos para un determinado umbral. Se puede observar que en un texto en lenguaje natural el orden máximo óptimo es 2. Los resultados producidos indican que en un texto en lenguaje natural no es muy beneficioso utilizar órdenes máximos elevados. Esto se debe a que utilizar un orden mayor supone almacenar un

[1] http://trec.nist.gov/

[2] Obtenido de http://pizzachili.dcc.uchile.cl/texts.html

[3] http://www.gutenberg.org/ebooks/

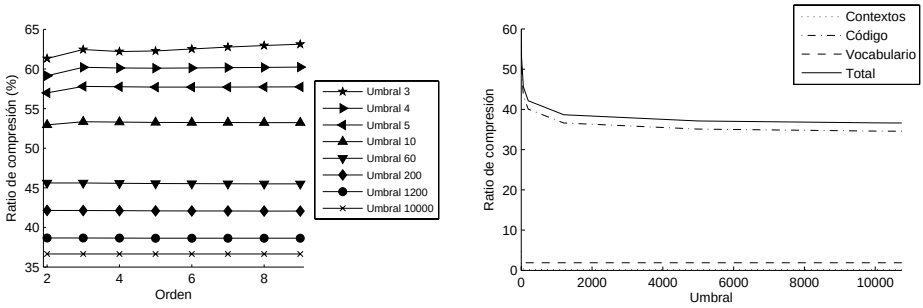


Figura 2. Ratios de compresión para un texto en lenguaje natural con diferentes umbrales y órdenes

mayor número de contextos y, para que compense el incremento de almacenamiento producido, la información disponible debe producir una mejora en la compresión que implique un *código* mucho más reducido. Si las frases de esta nueva longitud máxima no se repiten mucho, incrementar este orden supone la mayoría de las veces introducir un símbolo de escape y posteriormente codificar en el orden inferior del mismo modo que se hacía previamente, produciendo un *código* peor debido al coste de emisión del símbolo de escape.

Una vez fijado el orden óptimo, se procedió a analizar la cantidad de contextos que se debían almacenar (determinado por el umbral). La Gráfica derecha de la Figura 2 muestra los ratios de compresión para los órdenes máximos probados empíricamente para el texto *English50*. La gráfica indica en el eje *x* el umbral y, en el eje *y*, el ratio de compresión. En ella se muestran cuatro curvas: tres de ellas representan la contribución al ratio de compresión de los *contextos*, del *código* y del *vocabulario*, y una cuarta indica la suma de las tres anteriores, y por tanto, el ratio de compresión final.

Se puede observar cómo el espacio ocupado por los contextos decrece a medida que aumenta el umbral, debido a que ello implica que se almacenen un menor número de contextos, mientras que el vocabulario permanece constante. El tamaño del texto comprimido también desciende a medida que aumenta el umbral. En principio, se puede pensar que debería aumentar, ya que se dispone de menos información sobre los contextos y símbolos, pero la forma de codificar los símbolos de escape hace que en los umbrales bajos (en los que se eliminan un menor número de contextos), los símbolos de escape tengan una frecuencia más baja asignada, produciendo una peor codificación de estos símbolos que repercute en el texto comprimido final. La suma de las tres curvas se vuelve óptima en los umbrales superiores, por lo que se puede concluir que los resultados óptimos se obtienen para un umbral de 10000 o superior con un orden máximo de 2, resultando un ratio del 36,61 %.

Una vez calculados los valores óptimos para nuestro compresor, comparamos su comportamiento con otros compresores conocidos. Los compresores utilizados en la comparativa fueron el ya mencionado PPMDI, ETDC [2,3], gzip [4] y el compresor bzip2

[4] Puede encontrarse en el sitio Web http://www.gzip.org/

[5]. PPMDI fue probado con orden máximo 0 y 9. Por otra parte, el compresor gzip se utilizó con las opciones de ejecución más rápida (opción 1) y más eficiente, pero más lenta (opción 9), mientras que el compresor bzip2 se probó con los tamaños de bloques extremos (100k en la opción 1 y 900k en la opción 9 más rápido o mejor compresión respectivamente).

La Tabla 1 muestra los resultados obtenidos. En la primera columna se indica el nombre del corpus, mientras que las restantes muestran los ratios de compresión obtenidos con los diferentes compresores.

Los resultados obtenidos con SWPPM son competitivos, mejorando cualquier ejecución de gzip y de ETDC y con resultados próximos a la ejecución óptima de bzip2.

TEXTO	PPMDI		BZIP2		GZIP		ETDC	SWPPM
	Op. 0	Op. 9	Op. -1	Op. -9	Op. -1	Op. -9		O2,U10000
FT92	29,40 %	23,42 %	32,38 %	27,10 %	42,59 %	36,39 %	32,85 %	29,87 %
ZIFF	26,48 %	21,77 %	39,66 %	32,98 %	39,66 %	32,98 %	33,81 %	31,02 %
FT93	27,66 %	21,68 %	30,62 %	25,32 %	40,23 %	34,12 %	32,91 %	29,09 %
FT94	27,61 %	21,68 %	30,53 %	25,27 %	40,24 %	34,12 %	32,86 %	28,81 %
AP	30,34 %	23,33 %	33,27 %	27,25 %	43,66 %	37,23 %	32,93 %	30,05 %
FTALL	28,20 %	22,24 %	31,15 %	25,98 %	40,99 %	34,85 %	32,56 %	28,37 %

Tabla 1. Comparativa de diferentes compresores para textos semiestructurados

6. Conclusiones

En este trabajo hemos presentado un nuevo compresor PPM orientado a palabras (SWPPM), que se diferencia de las otras versiones de la familia PPM, en la utilización de palabras como símbolos a comprimir en lugar de caracteres.

Tras varios diseños preliminares, se creó una variante que llamamos SWPPM que sólo considera los contextos con una frecuencia superior a un umbral dado. El compresor propuesto fue probado experimentalmente obteniendo ratios de compresión del 28 %-31 %. La comparación frente a otras técnicas existentes permitió ver que el SWPPM comprime más que técnicas como el *gzip* (en todas sus variantes) y los compresores semi-estáticos orientados a palabras más utilizados en la actualidad (dense codes). En general, el SWPPM comprime un poco menos que el *bzip2*, y se ve superado por el *PPMDI*.

Los resultados prometedores obtenidos dejan la puerta abierta a futuras optimizaciones sobre la base del trabajo realizado, como por ejemplo, la mejora de la compresión de estructuras auxiliares que permiten mantener un mayor número de contextos durante la compresión. También estamos trabajando buscando una reducción del tamaño de los contextos en disco, realizando la compresión final con otras técnicas (ppm, gzip) en lugar de Huffman.

[5] Disponible en http://www.bzip.org/

Otra línea de investigación consiste en aprovechar la información que proporciona el hecho de emitir un símbolo de escape para obtener una mejor estimación de las probabilidades usadas al codificar en el orden inferior. Además, el filtro de contextos elimina aquellos elementos con una menor frecuencia porque es probable que se usen menos veces, pero es posible que algunos de los restantes no se utilicen, ya que a los elementos de órdenes inferiores sólo se accede cuando no existe el orden superior. Se propone por tanto realizar una pasada intermedia, previa al proceso de compresión, en la que se simula el proceso de codificación marcando el número de veces que se utiliza cada elemento y los símbolos de escape emitidos en cada contexto. Así, se ahorraría el espacio de almacenamiento de los elementos y se ajustarían mejor las frecuencias de los símbolos.

Referencias

1. T. C. Bell, I. H. Witten, and J. G. Cleary. *Text compression.* Prentice Hall, Englewood Cliffs, N.J., 1990.
2. N. R. Brisaboa, A. Fariña, G. Navarro, and J. R. Paramá. Lightweight natural language text compression. *Information Retrieval*, 10(1):1–33, 2007.
3. N. R. Brisaboa, E. L. Iglesias, G. Navarro, and J. R. Paramá. An efficient compression code for text databases. In *Proc. 25th European Conf. on IR Ressearch (ECIR'03) - LNCS 2633*, pages 468–481, 2003.
4. J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32:396–402, 1984.
5. D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.
6. A. Moffat. Word-based text compression. *Softw. Pract. Exper.*, 19(2):185–198, 1989.
7. A. Moffat, R. Neal, and I. H. Witten. Arithmetic coding revisited. In *Proc. IEEE Data Comp. Conf.*, pages 202–211. IEEE Computer Society Press, 1995.
8. A. Moffat and A. Turpin. *Compression and coding algorithms*. Kluwer Academic Publishers, Boston, London, 2002.
9. Turpin A. Moffat, A. Fast file search using text compression. In *Proc. 20th Australian Computer Science Conference*, pages 1–8, 1997.
10. E. Moura, G. Navarro, N. Z., and R. Baeza-Yates. Fast and flexible word searching on compressed text. *ACM TOIS*, 18(2):113–139, 2000.
11. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

Reducción del Tamaño del Índice en Búsquedas por Similitud sobre Espacios Métricos*

Luis G. Ares, Nieves R. Brisaboa, María F. Esteller,
Óscar Pedreira y Ángeles S. Places

Laboratorio de Bases de Datos, Facultade de Informática, Universidade da Coruña
Campus de Elviña s/n, 15071 A Coruña
{lgares,brisaboa,mfesteller,opedreira,asplaces}@udc.es

Resumen En problemas de recuperación de información se plantea frecuentemente la realización de búsquedas por similitud sobre espacios métricos. Se utilizan funciones de distancia costosas sobre grandes bases de datos, que requieren un equilibrio entre la cantidad de información almacenada en el índice y el número de evaluaciones de la función de distancia. Las soluciones adoptadas son métodos basados en pivotes y métodos basados en clustering. Los primeros obtienen menor número de comparaciones, pero presentan necesidades de espacio mucho mayores, por lo que se han propuesto alternativas que pretenden reducir esa cantidad de espacio. En este trabajo analizamos la utilidad de los pivotes en función de su proximidad al objeto, y como consecuencia, proponemos un nuevo método basado en pivotes que requiere una cantidad de espacio semejante a la que necesitan los métodos basados en clustering. Ofrecemos resultados experimentales que demuestran su idoneidad cuando se utiliza la misma cantidad de memoria.

1 Introducción

La búsqueda por similitud es una operación que está cada vez más presente en un gran número de aplicaciones que realizan el tratamiento de tipos de datos complejos, como es el caso de datos semiestructurados, información biológica y bases de datos multimedia. La recuperación basada en contendido es un problema habitual en estos escenarios, siendo la búsqueda por similitud una de las estrategias más adecuadas para solucionarlo. Un ejemplo de búsqueda por similitud consiste en obtener la imagen más semejante a una dada, pero su aplicación se extiende a una gran variedad de áreas, como los editores de texto y los grandes buscadores, donde se requiere encontrar palabras que se asemejan a otras para obtener resultados ante criterios de búsquedas o para efectuar comprobaciones sintácticas, el reconocimiento de patrones donde se realiza una detección y catalogación de documentos, imágenes o sonidos que presentan diversas similitudes, y la bioinformática donde se necesita reconocer y estudiar secuencias de ADN.

El problema de la búsqueda por similitud puede formalizarse mediante el concepto de *espacio métrico* que es un par (X, d) formado por un *universo* de objetos válidos X

* Este trabajo ha sido parcialmente financiado por el Ministerio de Educación y Ciencia (PGE y FEDER, ref. TIN2006-15071-C03-03) y por la Xunta de Galicia (ref. 2006/4)

y una función de distancia $d : X \times X \longrightarrow \mathbb{R}^+$ que cumple tres propiedades fundamentales: ser *estrictamente positiva* ($d(x, y) \geq 0$, y $d(x, y) = 0 \Leftrightarrow x = y$), ser *simétrica* ($d(x, y) = d(y, x)$) y verificar la *desigualdad triangular* ($d(x, y) \leq d(x, z) + d(z, y)$). La *base de datos* o *colección de objetos* sobre la que se realizan las búsquedas es un subconjunto finito $U \subseteq X$ de tamaño $n = |U|$.

Una *consulta* se expresa mediante un objeto a consultar $q \in X$ y un criterio de proximidad a ese objeto. El *conjunto resultado* es el conjunto de objetos de la colección que cumplen el criterio. La operación principal es la *búsqueda por rango* en la que se recuperan los objetos que están a una distancia de q menor o igual que un valor dado r. Otras operaciones pueden implementarse a partir de la anterior[1], destacando la *búsqueda de los k–vecinos más próximos*, que recupera los k objetos más similares a q.

Un espacio vectorial es un caso particular de espacio métrico en el que cada objeto está formado por un número de coordenadas reales. En este caso, como función de distancia d podría utilizarse alguna de la familia L_p definida sobre $\mathbb{R}^l$ como $L_p(x, y) = (\sum_{1 \leq i \leq l} |x_i - y_i|^p)^{1/p}$. Por ejemplo L_2 es la distancia euclídea. En una colección de cadenas de caracteres se puede utilizar la distancia de edición, que es el número de inserciones, borrados o sustituciones necesarias para transformar una cadena en la otra, teniendo así otro ejemplo de espacio métrico.

La búsqueda por similitud puede implementarse de forma trivial comparando el objeto a consultar con la totalidad de los objetos de la base de datos. Pero esta alternativa no es viable en términos prácticos, debido a que comparar los objetos supone un elevado coste computacional, máxime si la base de datos tiene un gran número de objetos. Por ello se han desarrollado métodos que construyen índices sobre la colección, y reducen el número de comparaciones entre objetos, utilizando la desigualdad triangular. De esta forma se pueden descartar elementos, sin compararlos directamente con la consulta.

Aunque la reducción en el número de evaluaciones de la función de distancia es el principal objetivo de estos métodos, hay otros factores de interés que afectan al rendimiento global del coste de la búsqueda, como son las necesidades de espacio para almacenar el índice (ya sea en memoria primaria o secundaria) y el tiempo de CPU adicional requerido para la carga y procesamiento del mismo. En unos casos los métodos son estáticos, en los que la colección no puede crecer una vez creado el índice; en otros son dinámicos y se adaptan a operaciones de inserción en una base de datos inicialmente vacía.

Los métodos de acceso en espacios métricos pueden clasificarse en dos grupos: los basados en clustering y los basados en pivotes[1]. Los métodos basados en clustering particionan el espacio y tratan de descartar regiones enteras durante la búsqueda. Los métodos basados en pivotes almacenan las distancias precalculadas de cada objeto en la base de datos a un conjunto de pivotes, y estas distancias se utilizan durante la búsqueda para descartar objetos del conjunto resultado. En los métodos basados en clustering, el índice suele ser una lista o un árbol que representa la partición del espacio. Necesitan una pequeña cantidad de memoria para almacenar el índice, siendo el tiempo extra de CPU también muy pequeño. Los métodos basados en pivotes superan a los métodos basados en clustering en términos de cálculos de distancia, a cambio de necesitar cantidades de espacio muy significativas para almacenar las distancias precalculadas.

En trabajos anteriores se han estudiado diferentes estrategias para reducir la cantidad de espacio utilizado por los métodos basados en pivotes, al tratar de conservar su eficacia para descartar objetos del conjunto resultado. Puede hacerse, fundamentalmente, almacenando menos información en el índice o utilizando menor precisión. En ambos casos, la pérdida de información implica una reducción de eficacia en el proceso de descarte de objetos.

En este trabajo, analizamos la eficacia de la selección de pivotes para cada objeto en la base de datos, y la utilidad de almacenar las distancias asociadas. Estudiamos también el efecto del conjunto de pivotes iniciales en el rendimiento del índice. Basándonos en este análisis, se propone un nuevo método basado en pivotes que necesita una cantidad de espacio muy cercana o incluso menor, que la utilizada en algoritmos basados en clustering. Nuestra evaluación experimental muestra que la propuesta representa una estrategia competitiva, cuando se usa la misma cantidad de espacio.

El resto del documento está organizado de la siguiente manera: la sección 2 describe las propuestas anteriores para reducir la cantidad de espacio en los métodos basados en pivotes, y las técnicas para la selección efectiva de los pivotes. La sección 3 se centra en los objetivos de este trabajo y describe nuestra configuración experimental. La sección 4 presenta nuestro análisis de la eficacia de los pivotes. La sección 5 presenta el nuevo método que proponemos en este artículo y su evaluación experimental. Por último, la sección 6 presenta las conclusiones de este trabajo y posibles líneas de trabajo futuro.

2 Antecedentes y Trabajos Relacionados

Sea (X, d) un espacio métrico y $U \subseteq X$ una base de datos o colección de objetos de tamaño $n = |U|$. Los métodos basados en pivotes seleccionan un conjunto de m objetos $P = \{p_1, \ldots, p_m\}$ de la base de datos, denominados pivotes. En la fase de indexación, se calculan las distancias $d(x_i, p_j)$ de los objetos en la base de datos a los pivotes, y se almacenan en una estructura de datos apropiada. Dada una consulta (q, r), el objeto de consulta q se compara con los pivotes para obtener las distancias $d(q, p_j)$. Por abuso de lenguaje se utiliza indistintamente *distancia de un elemento al objeto de consulta* y *distancia de un elemento a la consulta*. Para cada elemento $x_i \in U$, podemos obtener una cota inferior de la distancia a la consulta, usando la desigualdad triangular. El objeto se descarta del conjunto resultado, sin compararlo con la consulta, si $d(x_i, q) \geq |d(x_i, p_j) - d(p_j, q)| > r$ para algún pivote $p_j \in P$. La complejidad de la búsqueda es la suma de la complejidad interna (comparaciones de la consulta con los pivotes) y de la complejidad externa (comparación de la consulta con los objetos que no resultan descartados).

Los métodos basados en pivotes se diferencian en la información que almacenan y en las estructuras de datos que utilizan. Así AESA [2] y LAESA [3] usan una tabla para almacenar las distancias de los objetos a los pivotes, mientras que otros como FQT [4], FQA [5] o VPT [6] usan estructuras de árbol.

Dos factores relacionados con estos métodos son de gran importancia en el rendimiento global de la búsqueda. El primero se refiere a los requerimientos de espacio del índice, frecuentemente muy elevados, que conllevan además un tiempo de CPU adicional, necesario para la carga y procesamiento de la información almacenada en el índice.

Este es uno de los principales inconvenientes de los métodos basados en pivotes. El otro factor es la elección efectiva de los pivotes. La mayoría de los métodos seleccionan los pivotes al azar, pero se ha demostrado que el conjunto de pivotes determinado afecta al rendimiento [7], ya que la distribución de cada pivote con respecto a los demás, y al resto de los objetos en la base de datos, determina la capacidad del índice para descartar elementos del conjunto resultado. Ambos factores interactúan al considerar el número de pivotes elegidos, puesto que cuanto mayor sea el número de pivotes, mayores serán las necesidades de espacio del índice.

Los trabajos relacionados se han centrado en mejorar estos factores, intentado mantener la eficacia en el proceso de descarte de objetos.

2.1 Reducción de los Requerimientos de Espacio del Índice

Se han desarrollado varias estrategias para reducir el espacio del índice en los métodos basados en pivotes:

- Degradar el rango: Consiste en almacenar las distancias de los objetos a los pivotes con una precisión menor, con lo que se reduce el espacio necesario para el índice. Trabajando con distancias continuas -en el caso de distancias discretas es directo-, el rango de posibles valores se divide en varios intervalos y el índice almacena el intervalo en el que se encuentra cada distancia. Las distancias son ahora menos precisas, con lo que la cantidad de objetos desechados será menor. Este es el planteamiento utilizado por VPT [6], y puede aplicarse fácilmente en índices como FQA [5]. BAESA [8] propone algo semejante sobre un índice AESA [2] en el que el rango de distancias está dividido en 2^k intervalos, por lo que cada una de las $n \times n$ distancias de AESA se almacena utilizando k bits.
- Reducir el nivel de granularidad del cubo: Está orientada a índices con una estructura arbórea. Supone interrumpir la creación de subárboles, cuando se ha alcanzado un número reducido pero suficiente de elementos. El tamaño de la lista de candidatos aumenta, ya que determinadas áreas no forman parte del índice, pero este almacena menos distancias, con la consiguiente reducción de espacio.
- Reducir el ámbito de acción de los pivotes: Frente a métodos que almacenan las distancias de todos los objetos de la base de datos a cada uno de los pivotes, como AESA [2], LAESA [3] y SSS [9], esta alternativa propone guardar las distancias de cada pivote a un subconjunto de objetos en la base de datos. De esta forma se reduce el ámbito de actuación del pivote. El índice almacena así menos distancias, pero se reducen las posibilidades de descartar un objeto del conjunto resultado. KVP [10] es un ejemplo de esta estrategia.

2.2 Elección Efectiva de los Pivotes

La elección de un conjunto de pivotes que sea realmente efectivo, junto a determinar el número más favorable de pivotes, son factores que influyen en el rendimiento de la búsqueda. Si incrementamos el número de pivotes, aumentan las posibilidades de descartar objetos, pero a costa de aumentar la complejidad interna y también los requerimientos de espacio del índice. Por otra parte, los datos experimentales muestran que el número óptimo de pivotes depende de las características del espacio métrico.

Diversos trabajos realizan propuestas diferentes para determinar la eficacia de un conjunto de pivotes. En [3] y [6] se indica que los pivotes más adecuados deberían estar lejos unos de otros, y también del resto de objetos de la base de datos. En [7] se propone un criterio formal para comparar la eficacia de dos conjuntos de pivotes del mismo tamaño.

Estos trabajos están orientados a determinar la eficacia de un conjunto de pivotes, para la totalidad de los objetos de una base de datos, y no estudian cuál es el mejor pivote para un objeto determinado. Celik [10] muestra empíricamente que dado un objeto y una consulta, los mejores pivotes son aquellos que están o muy cerca o muy lejos del objeto, aunque en [11] menciona que es posible encontrar distribuciones para las que los pivotes cercanos y lejanos, no serían la mejor elección.

Se han propuesto varias técnicas para seleccionar pivotes, entre ellas MaxMin [12] que maximiza la distancia mínima entre pivotes, Spacing [13] que selecciona pivotes con una correlación mínima y que maximizan la distancia entre objetos, Incremental [7] que elige de forma incremental un conjunto de pivotes que maximizan un criterio para comparar dos conjuntos de pivotes. Sparse Spatial Selection (SSS) [9] selecciona un conjunto de pivotes de forma dinámica, determinando si un elemento será o no un pivote, en el momento en el que se introduce en la base de datos. Considera que un nuevo elemento está lo suficientemente lejos, si su distancia a los pivotes es mayor que $M\alpha$, siendo M la distancia máxima entre dos objetos cualesquiera, y α una constante relacionada con la densidad de pivotes en el espacio. Con este método adaptativo, se garantiza que el número de pivotes depende de la complejidad del espacio, y no del número de objetos.

3 Objetivos de este Trabajo

En este artículo nos planteamos los siguientes objetivos, referentes a los métodos basados en pivotes para la búsqueda por similitud en espacios métricos:

- Analizar la eficacia de los pitoves para cada objeto de la base de datos. Para ello realizamos un análisis empírico con diferentes colecciones de datos.
- Basándonos en los resultados del análisis anterior, reducir en lo posible el número de distancias almacenadas en el índice, para cada objeto de la base de datos.

En los experimentos hemos utilizado varias colecciones de datos, tanto reales como sintéticos, disponibles en la *Metric Spaces Library* [14]: UV08, UV10, UV12 y UV14 son colecciones de 100.000 vectores sintéticos con distribución uniforme y de dimensión 8, 10, 12 y 14 respectivamente. English es una colección de 69.069 palabras extraídas del diccionario de inglés, que se comparan utilizando la distancia de edición. Nasa es una colección de 40.150 imágenes extraídas de los archivos de la NASA, representadas por vectores de dimensión 20 y comparadas usando la distancia euclídea.

Las evaluaciones se centraron en la búsqueda por rango. En English trabajamos con $r = 2$. Para las colecciones de vectores, el radio de búsqueda se determinó en una media del $0,01\%$ de la base de datos, para cada consulta.

4 Análisis de la Eficacia de los Pivotes

Consideramos dos cuestiones relativas a la selección de los pivotes más eficaces para cada uno de los objetos de la base de datos: en primer lugar, el conjunto inicial de pivotes, entre los que se eligen los que resultan más prometedores, o sea, los que se espera que obtengan un mejor resultado, de cara a los descartes, en la evaluación de una consulta; en segundo lugar, la eficacia de los diferentes pivotes para cada objeto.

Celik [10] muestra que los pivotes más cercano y más lejano a un objeto son los más prometedores. Esto nos lleva a necesitar un método de selección de pivotes que garantice que los pivotes estén bien distribuidos en el espacio, lo que hará que para cada objeto haya pivotes tanto cercanos como lejanos. Estas condiciones las cumple SSS [9], mientras que otros métodos, en particular los que eligen los pivotes de manera aleatoria, no garantizan que cada objeto disponga de pivotes realmente cercanos o lejanos. En el peor de los casos, todos los pivotes pueden estar a la mitad de la distancia máxima al objeto. En este trabajo utilizamos SSS como alternativa de selección de pivotes.

La siguiente cuestión a dilucidar es que, entre los pivotes seleccionados con SSS, ¿cuáles son las características de los que resultan más útiles para descartar un objeto? A raíz de los resultados de Celik [10], sabemos que los pivotes deben ser cercanos o lejanos al objeto. Pero, ¿cuántos pivotes cercanos y lejanos necesitamos para cada objeto, y que a la vez sean eficaces?

La tabla 1 muestra el número de pivotes ("Piv."), espacio ("Espac.", en MB), y evaluaciones de la función de distancia ("Eval. d") obtenidas al utilizar:

- SSS, almacenando todas las distancias de cada objeto a cada pivote ("SSS (α óptimo)").
- Una selección aleatoria de pivotes, almacenando todas las distancias de cada objeto a cada uno de los pivotes ("Random").
- SSS, almacenando solo las distancias a los dos pivotes más cercanos y a los dos más lejanos a cada objeto ("SSS ($\alpha = 0,25; 4\ dist$)").
- SSS, almacenando solo las distancias al pivote más cercano y al más lejano a cada objeto ("SSS ($\alpha = 0,25; 2\ dist$)").

Colec.	Random			SSS (α óptimo)			SSS (α = 0,25; 4 dist.)			SSS (α = 0,25; 2 dist.)		
	Piv.	Espac.	Eval. d	Piv.	Espac.	Eval. d	Piv.	Espac.	Eval. d	Piv.	Espac.	Eval. d
UV08	85	29,1828	211,78	53	18,1963	141,43	494	2,7485	1231,66	494	1,3752	3094,13
UV10	190	65,2321	468,23	176	60,4255	367,14	1461	2,7522	3011,61	1461	1,3789	5891,64
UV12	460	157,9303	998,13	250	85,8317	645,08	4303	2,7630	6803,09	4303	1,3897	9739,49
UV14	1000	343,3266	2077,44	491	168,5734	1381,64	10000	2,7848	14229,20	10000	1,4115	18160,91
English	200	27,5696	443,85	212	45,0553	354,89	3100	1,9089	7931,30	3100	0,9604	12373,45
Nasa	77	10,6143	276,34	55	7,4438	168,62	871	1,1061	1308,28	871	0,5547	1958,34

Tabla 1. Coste de la búsqueda con diferentes estrategias al seleccionar un conjunto inicial de pivotes.

Lo obtenido es casi una reproducción de los resultados de Celik [10], aunque como podemos observar, el método utilizado para seleccionar el conjunto inicial de pivotes tiene una gran incidencia en la eficacia, mientras que el espacio se ha reducido considerablemente. Por ejemplo, en el caso de UV12, el número de evaluaciones al utilizar

dos pivotes es de aproximadamente 15 veces el número de comparaciones cuando se utilizan todos los pivotes, pero el espacio necesario al utilizar todos los pivotes es de aproximadamente 61 veces el espacio necesario cuando se utilizan solo dos. En el caso de UV14, una colección más compleja, la diferencia es aún mayor: el número de comparaciones cuando se utilizan dos pivotes es de aproximadamente 13 veces el número de evaluaciones necesarias cuando se utiliza la totalidad de los pivotes, pero el espacio utilizando todos los pivotes es de aproximadamente 119 veces el necesario utilizando solo dos. Podemos observar que en las colecciones más complejas, como UV14, English o Nasa, al pasar de 4 a 2 pivotes la eficacia se reduce un 25%, mientras que el espacio se reduce un 50%. Es decir, la pérdida de eficacia en evaluaciones de la función de distancia, es mucho menor, concretamente la mitad, que la reducción de espacio. Por lo tanto, pensamos que reducir drásticamente el número de pivotes a solo 2, es factible si se desean rebajar las necesidades de espacio del índice.

Una segunda cuestión que debemos preguntarnos es si es más eficaz utilizar el pivote más cercano o el más lejano para cada objeto. La figura 1 muestra, para cada colección, los porcentajes de los objetos que no son descartados ("No descartados"), los descartados solo por su pivote más cercano ("Solo por el más cercano"), los descartados solo por su pivote más lejano ("Solo por el más lejano"), y los descartados por ambos a la vez ("Por ambos"). En la figura podemos ver que la utilización del pivote más cercano, origina que se descarte la mayoría de los objetos, aunque un gran porcentaje puede ser descartado por la utilización, tanto del pivote más cercano como del más lejano. Solo en unos pocos casos el objeto resulta descartado por su pivote más lejano y no por el más cercano.

Con el fin de analizar estos datos y entender en qué casos se descarta el objeto solo por su pivote más lejano, calculamos la media y la desviación estándar de la distribución de distancias entre los objetos de cada colección. Posteriormente, para los casos en el que los objetos solo fueron descartados por su pivote más cercano ("Más cercano"), solo por el más lejano ("Más lejano") y por ambos ("Ambos"), calculamos la distancia media de los objetos en cada grupo, a los pivotes más cercano y más lejano. Los resultados se muestran en la tabla 2. Las distancias de los objetos a los pivotes se muestran en puntuaciones estándar (z) para poder realizar su comparación. Los pivotes más cercano y más lejano, se han obtenido utilizando SSS con $\alpha = 0,25$.

Colec.	μ	σ	Más cercano		Más lejano		Ambos	
			$zd_{cercano}$	zd_{lejano}	$zd_{cercano}$	zd_{lejano}	$zd_{cercano}$	zd_{lejano}
UV08	1,5086	0,2452	-4,3989	0,9845	-4,1949	1,3923	-4,3989	1,4331
UV10	1,4032	0,2456	-3,7182	2,2671	-3,5147	2,6743	-3,7182	2,6743
UV12	1,2652	0,2450	-3,0008	3,5298	-2,7151	3,9788	-3,0416	3,9380
UV14	1,1244	0,2469	-2,3669	4,6804	-1,9214	5,0855	-2,4480	5,0450
English	8,3176	2,0260	-2,3335	4,6113	-1,9040	5,1591	-2,2742	4,9617
Nasa	1,2342	0,3424	-2,2611	3,1419	-2,0859	2,9083	-2,1735	2,7623

Tabla 2. Puntuaciones estándar de las distancias a los pivotes más cercano y más lejano.

Tomando la colección English como ejemplo, observamos que la distancia media entre dos palabras es $\mu = 8,31$, y la desviación típica es $\sigma = 2,02$. Hay que tener en cuenta que, aunque la distribución de distancias es normal, las puntuaciones estándar se calculan sobre distancias de los objetos descartados a sus pivotes correspondientes.

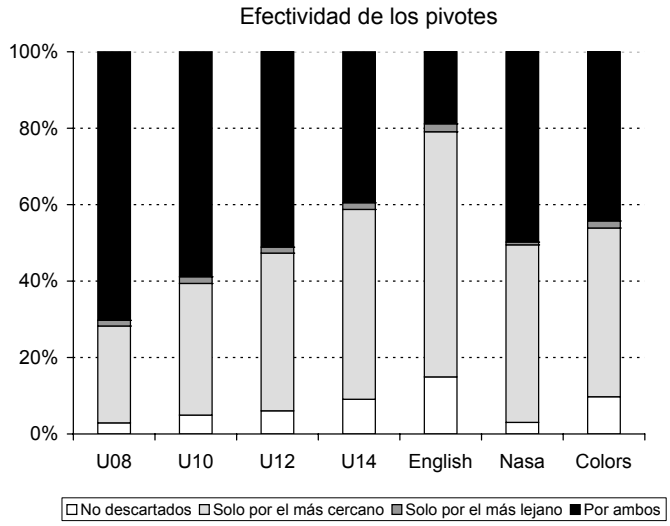


Fig. 1. Porcentajes de objetos descartados por el pivote más cercano, por el más lejano y por ambos.

Un objeto será descartado por su pivote más cercano, cuando la puntuación estándar de su distancia a ese pivote, es de aproximadamente $-2,33$. Sin embargo, cuando esa puntuación está cercana o es inferior a $-1,9$ y la puntuación estándar de la distancia al pivote más lejano es aproximadamente $5,1$, el objeto va a ser descartado únicamente por el más pivote más lejano. En las dos últimas columnas mostramos las puntuaciones estándar de las distancias a los pivotes más cercano y más lejano, para los objetos que pueden ser descartados por ambos.

Resulta curioso observar que, a medida que la complejidad de las colecciones sintéticas crece, hay pocas posibilidades de descartar el objeto mediante su pivote más lejano. Estos resultados son acordes con los obtenidos en la figura 1.

Estas puntuaciones nos dan un umbral para cada colección, que nos permite decidir, para cada objeto, si será el pivote más cercano o el más lejano, el que tenga más posibilidades de descartarlo.

Como conclusión, el pivote más cercano es el que tiene mayor capacidad para descartar un objeto, y solo en aquellos casos en los que no está suficientemente cerca, vale la pena tratar de utilizar el pivote más lejano, si realmente está lo suficientemente lejos. Las puntuaciones estándar de la distancia al pivote más cercano y al pivote más lejano, pueden darnos para cada colección, una guía de cuando utilizar el pivote más lejano. En caso de duda, porque la distancia de ambos podría estar por encima o por debajo de estos umbrales, la opción más fiable es usar el pivote más cercano.

5 Índice de Espacio Mínimo Basado en Pivotes

Basándose en los resultados de nuestro análisis empírico acerca de la eficacia de los pivotes, proponemos construir un índice en el que, para cada objeto almacenamos el identificador del pivote que se va a utilizar y la distancia del objeto a ese pivote. Dicho pivote será el más cercano, o el más lejano en aquellos casos en los que la distancia del objeto al más cercano no es lo suficientemente pequeña, y a la vez existe un pivote suficientemente lejano.

Podríamos ver el índice resultante como si se crease una partición de Voronoi sobre el espacio, almacenando para cada objeto, la partición a la que pertenece y la distancia al pivote central de la partición. Sin embargo, en nuestro caso, para los objetos que no están lo suficientemente cerca del centro de su partición (porque están distantes del resto de elementos de la colección o porque están muy cerca del límite de dos regiones), almacenamos información sobre su pivote más lejano, en lugar de sobre el más cercano.

5.1 Almacenamiento de Distancias con Menor Precisión

Ya que nuestro objetivo es reducir el espacio del índice, podemos almacenar los identificadores de los pivotes con $\log_2(m)$ bits, siendo m el número de pivotes, por lo que necesitarán menos de un byte en la mayoría de los casos. La distancia al pivote puede almacenarse con un número variable de bits, si permitimos perder alguna precisión.

Para evaluar nuestro método, nos referiremos a él como Índice de Pivote Único (UPI). La tabla 3 muestra dos configuraciones diferentes del método. En la primera ("UPI (4 bytes)"), los identificadores de los pivotes se almacenaron utilizando el número mínimo de bits, y para las distancias se usaron los 4 bytes necesarios para precisión flotante. En el segundo ("UPI (2 bytes)"), para los identificadores de los pivotes se utilizó también el número mínimo de bits, pero las distancias se almacenaron utilizando 2 bytes, es decir, con la mitad de precisión. Para cada opción se muestra el espacio utilizado por el índice en MB ("Espac.") y el número de evaluaciones de la función de distancia ("Eval. d"). Podemos ver en la tabla que cuando las distancias se almacenan con pérdida de precisión, la desventaja en el número de comparaciones es casi inexistente, mientras que el espacio se reduce aproximadamente a la mitad del necesario cuando se usa precisión flotante para las distancias.

Colec.	UPI (4 bytes)		UPI (2 bytes)	
	Espac.	Eval. d	Espac.	Eval. d
UV08	0,4311	3094,13	0,2594	3095,72
UV10	0,4348	5891,64	0,2631	5893,84
UV12	0,4456	9739,49	0,2740	9741,91
UV14	0,4673	18160,91	0,2957	18164,03
English	0,3083	12373,45	0,1897	12373,45
Nasa	0,1757	1958,34	0,1068	1958,72

Tabla 3. Espacio y evaluaciones de la distancia cuando se utiliza menor precisión.

El número de bits necesarios para almacenar las distancias, depende del rango de valores obtenido por la función de distancia. Por lo tanto, nuestro método puede ser parametrizado para guardar las distancias con un determinado número de bits, en función de la definición de la función de la distancia y de la tolerancia a la pérdida de precisión.

Colec.	SSS (α óptimo)		KVP (k=2)		UPI		Lista de Clusters	
	Espac.	Eval. d	Espac.	Eval. d	Espac.	Eval. d	Espac.	Eval. d
UV08	18,1963	141,43	1,3752	8724,62	0,2594	3095,72	0,3643	6139,21
UV10	60,4255	367,14	1,3789	13063,63	0,2631	5893,84	0,3710	11264,98
UV12	85,8317	645,08	1,3897	18955,94	0,2740	9741,91	0,3710	17273,55
UV14	168,5734	1381,64	1,4115	28502,26	0,2957	18164,03	0,3710	28253,04
English	45,0553	354,89	0,9604	18305,43	0,1897	8872,37	0,2651	7885,79
Nasa	7,4438	168,62	0,5547	2676,93	0,1068	1958,72	0,1427	2027,08

Tabla 4. Comparación con otros métodos.

5.2 Evaluación

Para comprobar la competitividad del método propuesto, lo comparamos con los siguientes métodos:

- SSS con el espacio necesario para su configuración óptima ("SSS (α óptimo)").
- KVP almacenando para objeto solo la distancia a sus pivotes más cercano y más lejano ($k = 2$) ("KVP ($k = 2$)").
- Lista de Clusters [15], puesto que se trata de un método representativo de la efectividad de los métodos basados en clustering ("Lista de Clusters").

Aunque SSS realiza un número mucho menor de evaluaciones, lo comparamos con nuestro método porque constituye una buena base de referencia para el número de evaluaciones de la distancia y para la utilización de espacio.

La tabla 4 muestra los resultados obtenidos. Nuestro método obtiene mejores resultados que KVP en todas las colecciones utilizando menos espacio para almacenar el índice. Frente a Lista de Clusters, nuestro método obtiene mejores resultados en términos de evaluaciones de la función de distancia, en todas las colecciones de vectores sintéticos y en Nasa, usando además menos espacio para el índice. En el caso de English, nuestro método necesita más evaluaciones de la función de distancia, pero requiere menos espacio para almacenar el índice. El número de evaluaciones de la función de la distancia es, por supuesto, superior a los obtenidos con SSS, pero la reducción del espacio necesario para el índice es muy importante. Por ejemplo, en el caso de UV14, SSS crea un índice de cerca de 168 MB, mientras que nuestro método solo necesita $0,37$ MB para ello. Estos resultados demuestran que nuestro método es una estrategia competitiva con las propuestas anteriores, cuando se usa la misma cantidad de espacio.

6 Conclusiones

En este trabajo, consideramos el problema de la reducción de los requerimientos de espacio en los métodos basados en pivotes, para la búsqueda por similitud en espacios métricos.

Presentamos un análisis empírico de la eficacia de los pivotes para cada objeto de la base de datos, estudiando el efecto de la elección del conjunto inicial de pivotes y la selección de los más prometedores para cada objeto.

Nuestros resultados muestran también que podemos almacenar para cada objeto, únicamente la distancia a su pivote más prometedor, siendo este el más cercano o el más lejano, y reducir la cantidad de espacio necesaria para el índice, a la que requieren los métodos basados en clustering.

Basándose en los resultados de este análisis, proponemos un nuevo método basado en pivotes que necesita una cantidad de espacio menor o muy cercana a la necesaria para los métodos basados en clustering. Nuestro método combina las estrategias de reducción de rango y de ámbito: almacenamos solo la distancia más prometedora para cada objeto, y para ello utilizamos la mitad de la precisión.

Nuestros resultados muestran que el método supone una estrategia competitiva frente a propuestas anteriores, cuando usamos la misma cantidad de espacio.

Como trabajo futuro, estamos estudiando la parametrización del número de bits usados para almacenar la distancia.

Referencias

1. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Computing Surveys **33**(3) (September 2001) 273–321 ACM Press.
2. Vidal, E.: An algorithm for finding nearest neighbors in (approximately) constant average time. Pattern Recognition Letters **4** (1986) 145–157 Elsevier.
3. Micó, L., Oncina, J., Vidal, R.E.: A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear pre-processing time and memory requirements. Pattern Recognition Letters **15** (1994) 9–17 Elsevier.
4. Baeza-Yates, R., Cunto, W., Manber, U., Wu, S.: Proximity matching using fixed-queries trees. In: Proc. of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM'94), Asilomar, C.A., Springer-Verlag (1994) 198–212
5. Chávez, E., Marroquín, J.L., Navarro, G.: Fixed queries array: A fast and economical data structure for proximity searching. Multimedia Tools and Applications **14**(2) (2001) 113–135
6. Yianilos, P.: Data structures and algorithms for nearest-neighbor search in general metric spaces. In: Proc. of the fourth annual ACM-SIAM Symposium on Discrete Algorithms (SODA'93), ACM Press (1993) 311–321
7. Bustos, B., Navarro, G., Chávez, E.: Pivot selection techniques for proximity searching in metric spaces. Pattern Recognition Letters **24**(14) (2003) 2357–2366 Elsevier.
8. Figueroa, K., Fredriksson, K.: Simple space-time trade-offs for aesa. In: Proc. 6th Workshop on Efficient and Experimental Algorithms (WEA 2007). LNCS 4525, Springer-Verlag (2007) 229–241
9. Brisaboa, N.R., Fariña, A., Pedreira, O., Reyes, N.: Similarity search using sparse pivots for efficient multimedia information retrieval. In: Proc. of the 8th IEEE International Symposium on Multimedia (ISM'06), San Diego, California, USA, IEEE Press (2006) 881–888

10. Celik, C.: Priority vantage points structures for similarity queries in metric spaces. In: Proc. of EurAsia-ICT 2002: Information and Communication Technology. Volume 2510 of Lecture Notes in Computer Science., Springer (2002)
11. Celik, C.: Effective use of space for pivot-based metric indexing structures. In: Proc. of Int. Workshop on Similarity Search and Applications (SISAP 2008), Cancún, México, IEEE Press (2008) 402–409
12. Vleugels, J., Veltkamp, R.C.: Efficient image retrieval through vantage objects. Pattern Recognition **35**(1) (2002) 69–80 Elsevier.
13. van Leuken, R.H., Veltkamp, R.C., Typke, R.: Selecting vantage objects for similarity indexing. In: Proc. of the 18th International Conference on Pattern Recognition (ICPR 2006), IEEE Press (2006) 453–456
14. SISAP: Metric spaces library (`http://sisap.org/metric_space_library.html`)
15. Chávez, E., Navarro, G.: A compact space decomposition for effective metric indexing. Pattern Recognition Letters **26**(9) (2005) 1363–1376

Reorganizing Compressed Text ⋆

Nieves R. Brisaboa[1], Antonio Fariña[1], Susana Ladra[1], and Gonzalo Navarro[2]

[1] Database Laboratory, University of A Coruña, Spain
{brisaboa,fari,sladra}@udc.es
[2] Department of Computer Science, University of Chile, Chile
gnavarro@dcc.uchile.cl

Key words: Word-based compression, searching compressed text, compressed indexing.

Recent research has demonstrated beyond doubts the benefits of compressing natural language texts using word-based statistical semistatic compression. Not only it achieves extremely competitive compression rates, but also direct search on the compressed text can be carried out faster than on the original text; indexing based on inverted lists benefits from compression as well.

Such compression methods assign a variable-length codeword to each different text word. Some coding methods (Plain Huffman and Restricted Prefix Byte Codes) do not clearly mark codeword boundaries, and hence cannot be accessed at random positions nor searched with the fastest text search algorithms. Other coding methods (Tagged Huffman, End-Tagged Dense Code, or (s, c)-Dense Code) do mark codeword boundaries, achieving a self-synchronization property that enables fast search and random access, in exchange for some loss in compression effectiveness.

In this paper, we show that by just performing a simple reordering of the target symbols in the compressed text (more precisely, reorganizing the bytes into a wavelet-tree-like shape) and using little additional space, searching capabilities are greatly improved without a drastic impact in compression and decompression times. With this approach, all the codes achieve synchronism and can be searched fast and accessed at arbitrary points. Moreover, the reordered compressed text becomes an *implicitly indexed* representation of the text, which can be searched for words in time independent of the text length. That is, we achieve not only fast sequential search time, but indexed search time, for almost no extra space cost.

We experiment with three well-known word-based compression techniques with different characteristics (Plain Huffman, End-Tagged Dense Code and Restricted Prefix Byte Codes), and show the searching capabilities achieved by reordering the compressed representation on several corpora. We show that the reordered versions are not only much more efficient than their classical counterparts, but also more efficient than explicit inverted indexes built on the collection, when using the same amount of space.

⋆ In *Proceedings of the 31st annual international ACM SIGIR conference*, pages 139-146, Singapore, July 2008, doi: http://doi.acm.org/10.1145/1390334.1390360

Exploiting Pipeline Interruptions for Efficient Memory Allocation

Josep Aguilar-Saborit[2], Mohammad Jalali[2], Dave Sharpe[2], and Victor Muntés-Mulero[1]

[1] DAMA-UPC, Computer Architecture Dept., Universitat Politècnica de Catalunya, Campus Nord UPC, C/Jordi Girona 1-3 08034 Barcelona, (Catalonia, Spain), vmuntes@ac.upc.edu

[2] IBM Toronto Lab, 8200 Warden Ave. Markham, ON L6G1C7, Canada {jaguilar, mjalali, dsharpe}@ca.ibm.com

Key words: Performance, Memory Allocation, Query post-optimization

The execution of a query in database management systems (DBMSs) is usually divided into different phases. Depending on the distribution of memory resources among the different internal operations, the execution time of a query can change drastically. This problem arises in very common scenarios for multi-join query processing, posing a serious problem for query optimization. If we just consider the join ordering problem, the search space of possible QEPs to solve a specific query grows exponentially to the number of joins in the query. The complexity of this problem is increased even further when we consider the use of different scan and join implementations, or the exploration of bushy trees instead of left-deep trees. Commercial DBMSs such as DB2 UDB or Oracle fall into static approaches where (i) the DBA sets the start-up parameters that the compiler gathers to compute the query execution plan, and (ii) the memory assigned to all the memory-intensive operations in the QEP is usually distributed equally. The main drawback of this approach is twofold: first, it relies on the expertise of the DBA; second, it is sub-optimal because it does not take into account that different operators in a QEP can have completely different memory needs.

As an alternative, research has been carried out in order to statically assign memory via rule-based approaches during compile time, or dynamically readjust the memory available for each operation during the execution of the query. Other approaches try to solve this problem from a cost-basis perspective during the compilation time of a query. However, they do not consider bushy trees, which are commonly used in commercial systems. In addition, they base their proposals on a theoretical cost function that differs significantly from those in real systems in terms of complexity, and makes them difficult to be applied in practice.

In our work, we tackle the problem of memory allocation at compile time among the operations in a QEP. Although most of the DBMSs follow a pipelined philosophy in order to execute the queries, a large number of common QEP operations interrupt the data flow by requiring reading at least one of the input data sets completely before starting to return results. The main idea of our work is to exploit the available memory of the system by reusing the memory released by these operations. Specifically, we focus on the hybrid hash join (HHJ) operation that, among all join algorithms, is characterized

by being memory-intensive and very dependent on the available memory to achieve good performance.

We study the coexistence of multiple operations competing for memory during the QEP processing. We propose a method to identify clusters of operations that compete for memory during the execution of a query and that allow for bushy trees. We also propose the *Memory Chunk Distribution* (MCD) algorithm. MCD is a method to distribute the available memory among the different operations in a cluster in order to achieve a better utilization of the available memory, independently of the complexity of the cost function. Finally, we implement our algorithm in DB2 UDB, and show that our memory allocation techniques improve the query execution performance compared with the current DB2 algorithm and other algorithms presented in the literature, both in terms of I/O and time.

Abstract of the paper included in the Proceeding of the 17th ACM conference on Information and knowledge management (CIKM), pp. 639-648, 2008.

Parte VIII

Sesión 7. Líneas de Producto

Towards security requirements management for software product lines: a security domain requirements engineering process

Daniel Mellado[1], Eduardo Fernández-Medina[2] y Mario Piattini[2]

[1] Tax Administration National Agency,
Madrid (Spain)
Daniel.Mellado@uclm.es
[2] University of Castilla La-Mancha, Alarcos Research Group – Institute of Information Technologies & Systems, Dep. of Information Technologies & Systems
Paseo de la Universidad, 4 13071 Ciudad Real (Spain)
{Eduardo.FdezMedina,Mario.Piattini}@uclm.es

Abstract. Security and requirements engineering are one of the most important factors of success in the development of a software product line due to the complexity and extensive nature of them, given that a weakness in security can cause problems throughout the products of a product line. The main contribution of this work is that of providing a security standard-based process for software product line development, which is an add-in of activities in the domain engineering. This process deals with security requirements from the early stages of the product line lifecycle in a systematic and intuitive way especially adapted for product line based development. It is based on the use of the latest security requirements techniques, together with the integration of the Common Criteria (ISO/IEC 15408) and the ISO/IEC 17799 controls into the product line lifecycle. Additionally, it deals with security artefacts variability and traceability, providing us with a Security Core Assets Repository. Moreover, it facilitates the conformance to the most relevant security standards with regard to the management of security requirements, such as ISO/IEC 27001 and ISO/IEC 17799. Finally, we will illustrate our proposed process by describing part of a real case study, as a preliminary validation of it.

Keywords: Product lines; Common Criteria; ISO/IEC 27001; Security requirement; Security requirements engineering.

Referencia completa de la publicación:

Autores: Mellado, D., Fernández-Medina, E., y Piattini, M.
Título: "Towards security requirements management for software product lines: a security domain requirements engineering process" *(accepted –5 march 2008, available online 8 march 2008)*
Referencia revista / libro: **Computer Standards and Interfaces**
Volumen: 30 Páginas, inicial: 361 final: 371 Fecha: Agosto-2008 ISSN: 0920-5489
JCR Factor de Impacto: 0.509 (en 2007)
Editorial (si libro): Elsevier Lugar de publicación: http://dx.doi.org/10.1016/j.csi.2008.03.004

Factor de Impacto: 0,509 (en 2007). **Trabajo relacionado "in Press"** en la misma revista: http://dx.doi.org/10.1016/j.csi.2006.04.002 . Caso de estudio sobre el trabajo propuesto en el artículo que proponemos para su divulgación, publicado en el congreso ARES'2009 (Factor de **aceptación: 25%**) "*Automated Support for Security Requirements Engineering in Software Product Line Domain Engineering*" (DOI 10.1109/ARES.2009.23): p. 224-231.

A Software Product Line Definition for Validation Environments

Belen Magro[1], Jennifer Perez[2] and Juan Garbajosa[2]

[1] Dimetronic
Belen.Magro -at- dimetronic.com
[2] Technical University of Madrid (UPM)
http://syst.eui.upm.es
jennifer.perez,jgs -at- eui.upm.es

Abstract. Agile methods have appeared as an attractive alternative to conventional methodologies. These methods try to reduce the time to market and, indirectly, the cost of the product through flexible development and deep customer involvement. The processes related to requirements have been extensively studied in literature, in most cases in the frame of conventional methods. However, conclusions of conventional methodologies could not be necessarily valid for Agile; in some issues, conventional and Agile processes are radically different. As recent surveys report, inadequate project requirements is one of the most conflictive issues in agile approaches and better understanding about this is needed. This paper describes some findings concerning requirements activities in a project developed under an agile methodology. The project intended to evolve an existing product and, therefore, some background information was available. The major difficulties encountered were related to non-functional needs and management of requirements dependencies.

Abstract of the paper included in the Proceeding of the 12th International Software Product Line Conference (SPLC), IEEE Computer Society, pp. 45-54, 2008

Actas JISBD 2009, pp.268, ISBN: 978-84-692-4211-7

Realizing Feature Oriented Software Development with Equational Logic: An Exploratory Study

Roberto E. Lopez-Herrejon[1], Jose Eduardo Rivera[2]

[1] Johannes Kepler University, Institute for Systems Engineering and Automation, Linz, Austria
roberto.lopez-herrejon@jku.at
[2] Dept. Lenguages y Ciencias de la Computacion, University of Malaga, Spain
rivera@lcc.uma.es

Abstract. A Software Product Line (SPL) is a family of related programs distinguished by the features they provide. Feature-Oriented Software Development (FOSD) is a paradigm that provides formalisms, methods, languages and tools for building variable, customizable and extensible software. FOSD has been successfully used to implement several SPL case studies. At the heart of FOSD is a feature algebra that drives the (de)composition of software artifacts. Current implementations of FOSD realize its algebra with a sophisticated combination of parser, compiler and scripting technologies. Though effective, such realizations depart from the algebraic nature of FOSD, limiting the algebra to the architectural description, not permeating down to the actual composition of the software artifacts. In this paper, we argue that equational algebra can help bridge the conceptual gap between FOSD concepts and their implementation. We used Maudeling, a model management tool, to implement the basic FOSD algebra. Maudeling is based on Maude (a high-performance logical framework) that provides support for equational logic. We applied our implementation to a product line of UML class diagrams. Our results suggest a potential for realizing FOSD concepts with this paradigm.

1 Introduction

A *Software Product Line (SPL)* is a family of related programs distinguished by the *features* (i.e. increments in program functionality relevant to stakeholders) they provide [1]. Different programs of an SPL provide different combinations of features. Extensive research has shown how SPL practices can improve factors such as asset reuse, time to market, or product customization and the important economical and competitive advantages they entail [2]. *Feature-Oriented Software Development (FOSD)* is a paradigm that provides formalisms, methods, languages, and tools for building variable, customizable and extensible software [1, 3, 4]. FOSD has been successfully used to implement several SPL case studies [4]. At the heart of FOSD is a feature algebra that drives the (de)composition of the software artifacts used throughout the software development life cycle [1].

At the time of writing, this algebra has been applied to several artifact types (e.g. source code, XML, grammars) and implemented with a sophisticated combination of parser, compiler and scripting technologies. Though effective, such realizations depart

Actas JISBD 2009, pp.269-274, ISBN: 978-84-692-4211-7

from the algebraic nature of FOSD, limiting the algebra to the architectural description, not permeating down to the actual composition of the software artifacts. In this paper, we argue that equational algebra can help bridge the conceptual gap between FOSD concepts and their implementation. We used Maudeling, a model management tool, to implement the basic FOSD algebra. Maudeling is based on Maude (a high-performance logical framework) that provides support for equational logic. As a first assessment effort we applied our implementation to a product line of UML class diagrams. Our results suggest a potential for realizing FOSD concepts with this paradigm.

2 Feature Oriented Software Development

Feature Oriented Software Development (FOSD) is a computational paradigm for program synthesis [5]. It has been successfully used for the development of SPLs in different domains [4]. The underlying principles, ideas, and mathematics of FOSD have evolved over almost two decades of continuous research. *Algebraic Hierarchical Equations for Application Design (AHEAD)* is a form of FOSD with a simple algebra that governs the composition of features [1, 5]. Features are expressed as tuples of *deltas* that correspond to refinements, extensions or interactions [5]. AHEAD defines operator $\bullet$ over the set of features F, and a program p as a composition of features:

$$\bullet : F \times F \rightarrow F \qquad\qquad p = f_n \bullet f_{n-1} \bullet \ldots \bullet f_2 \bullet f_1 \tag{1}$$

Operator $\bullet$ is associative, idempotent, non-commutative and has an identity ξ which represents the empty feature. AHEAD features are hierarchical structures, thus they can be understood as trees where the root is a feature while the children nodes are the deltas that constitute the feature. Feature composition can then be described as follows. Starting from the root of two trees, a pair of nodes is composed recursively when they have the same name, type, and path in their corresponding trees. At the leaf level, the exact nature of the composition depends on the artifact type and the specific semantics desired for it. For instance, in UML class diagrams, examples of node types are class, method, and field. Nodes without matching nodes at the other tree are simply copied to the resulting tree at the point where no match was found [1].

3 Maudeling

Maudeling is a model management tool, developed as an Eclipse plug-in, that provides several model operations such as model subtyping and difference [6]. Using ATL, Maudeling transforms Eclipse models to and from their Maude representation on which operations are performed and results are expressed. In this section, we informally present the Maude concepts necessary for understanding the paper; the interested reader is referred to its manual for more details [7].

Maude is a declarative language of the OBJ algebraic specification family [7]. It is implemented by an open source high-performance interpreter and compiler, and supported with a set of tools that help verifying certain program properties such as termination and coherence [7]. Maude has been successfully applied to several areas such

as programming languages for the development of program analysis tools and formal semantics specifications, distributed and concurrent systems for analysis of protocols, software engineering development methods, and more recently for Model-Driven Development [8–10].

In equational logic, computation is accomplised by using equations, from left to right, as simplification rules until reaching a *canonical form*, a term that cannot be further simplified. Maude's underlying equational logic is membership-equational logic, a Horn logic whose atomic sentences are equalities `t = t'`, and membership assertions of the form `t : S`, stating that a term `t` has sort (type) `S`. Some equations, like those expressing the commutativity of binary operators, are not terminating but nonetheless they are supported by means of operator attributes, so that Maude performs simplification modulo the equational theories provided by such attributes, which can be associativity (`assoc`), commutativity (`comm`), identity (`id`), and idempotence (`idem`). It is precisely this native support of operation attributes that brought our attention to Maude as an alternative to implement FOSD.

Models are structures of the form $\texttt{mm}\{\texttt{obj}_1\ \texttt{obj}_2\ \ldots\ \texttt{obj}_n\}$, where `mm` is the name of the metamodel, and $\texttt{obj}_i$ are the objects that constitute the model. An object is a record-like structure of the form $\texttt{<}\ \texttt{O}\ \texttt{:}\ \texttt{C}\ \texttt{|}\ \texttt{(}\ \texttt{a}_1\texttt{:v}_1\texttt{,}\ \ldots\ \texttt{,}\ \texttt{a}_n\texttt{:v}_n\ \texttt{)>}$, where `C` is the class (type) of the object, `O` is a quoted identifier, and $\texttt{a}_i\texttt{:v}_i$ are attribute-value pairs.

Maudeling categorizes object attributes in two metatype groups: @Attribute to hold the values of an object, and @Reference to hold references to other objects via their identifiers. The first group allows to represent data such as name while the second was used to represent the hierarchical structures of the feature trees.

For instance, the following object is the Maudeling representation of a UML class `Phone`. This object represents a UML class with 3 Maude attributes. The value of the first attribute holds the name of the class. The value of the second attribute is an ordered set of quoted identifiers whose corresponding objects represent the fields (attributes) of the class. Similarly, the value of the third attribute is an ordered set of quoted identifiers whose corresponding objects represent the operations of class.

$$\begin{array}{l} <' phone : Class@uml | (name@NamedElement@uml : "Phone", \\ ownedAttribute@StructuredClassifier@uml : ('number -, 'owner), \\ ownedOperation@Class@uml : ('placeCall -, 'computeCharges)) > \end{array} \quad (2)$$

Maudeling generates for each metaclass a sort and a constant of the corresponding sort; and uses Maude's subsorting support to implement class inheritance. Other properties of the metaclasses are expressed with equations defined over the sort constant that represents the metaclass [6].

4 FOSD Realization with Maudeling

We represent each node on a feature tree as a Maudeling object; thus a feature is a set of objects. The class of an object is the metaclass of the element it represents. For instance, the object in Equation (2) represents a class node with its corresponding metaclass `Class@uml`.

We implemented our FOSD algebra in two Maude modules [7]: `Kernel` and `ClassDiagram`. The `Kernel` module implements the core of the FOSD algebra as explained in Section 2. This module defines 72 operators and 158 equations in a total of 458 LOC. Its main operator `dot` leverages Maude's native support of operator attributes to express the properties of our algebra. The majority of the operators provide support for manipulating objects and their attributes. A distinctive characteristic of this module is that all operators were defined exclusively in terms of Maudeling's metatypes which makes the algebra applicable to any model represented in Maudeling.

As mentioned in Section 2, the specific composition semantics of • operator depends on the type of artifact. Because of this reason, we implemented the composition details particular to class diagrams in a separate module `ClassDiagram`. This module defines 12 operators and 23 equations in a total of 98 LOC. Method composition is an example of specific composition semantics of class diagrams where two methods are composed if they have the same signature. In this case, the module effectively overrides the default composition when two method nodes are reached.

Modules can then extend and refine the operators in the `Kernel` module. This modular implementation can facilitate the development of the basic algebra for other artifact types. On a first stage, we plan to extend our work to state machines and sequence diagrams. Later, we intend to extend the work to other metamodels and more sophisticated algebras such as [3].

5 Case Study

We used the *Graph Product Line (GPL)* [11] as case study to evaluate our implementation of FOSD basic algebra. The features of this product line are basic graph algorithms and data structures. A GPL program is a combination of different algorithms implemented on different data structures. There are implementations of GPL available in several programming languages but there are no UML class diagrams of its features. Our approach was to take an extended Java-based implementation and develop a translator from Java to Maudeling, using ANTLR to parse the source files and Velocity template engine to generate Maudeling code. This version of GPL has 27 features amounting to a total of 684 UML elements. Table 1 shows each GPL feature and its corresponding number of UML objects generated in Maudeling representation. We composed 3 representative examples of GPL programs to help us gauge the performance of our algebra implementation. These compositions were executed on a Windows laptop, running at 1.8 GHz with 4GB RAM. For each example we gather the number of objects in Maudeling representation, the number of rewrites performed, and the composition time in miliseconds averaged over 3 runs. The results are summarized in Table 2. We found that for these examples the composition time does not significantly exceeds a second even for the largest program GPL3. As expected, the number of rewrites increases as more features are composed. Certainly these results only provide us with glimpse of the performance of our approach. So as part of our future work we plan to evaluate larger and more complex examples of class diagrams and other models.

Feature Name	Objs	Feature Name	Objs	Feature Name	Objs
Base	25	MSTKruskal	10	UndirectedWithEdges	76
BFS	26	MSTPrim	11	UndirectedWithNeighbors	69
Connected	17	Number	13	UnWeighted	1
Cycle	28	OnlyVertices	1	Weighted	25
DFS	24	StronglyConnected	24	WeightedOnlyVertices	23
Directed	1	TestProg	4	WeightedWithEdges	17
DirectedOnlyVertices	56	Transpose	5	WeightedWithNeighbors	33
DirectedWithEdges	69	UnDirected	1	WithEdges	1
DirectedWithNeighbors	60	UndirectedOnlyVertices	63	WithNeighbors	1

Table 1. GPL Features and Number of Elements

6 Related Work

There is a significant amount of related literature. Thus we focus on the research that most closely relates to our work and divide them in three categories.

	GPL 1	GPL 2	GPL 3
Num Features	8	10	12
Num Objects	140	183	241
Composed Objects	124	155	173
Rewrites	74178	138090	206941
Time msec	402	691	1038

Table 2. GPL Examples Statistics

Algebras for FOSD. Our previous work has shown the applicability of an algebraic representation to describe model composition in *use case slices*, an Aspect-Oriented modeling techniques based on UML diagrams, when used for SPL modeling [12].

Model Composition and Management. Herrmann et al. [13] propose an algebraic approach to describe the semantics of model composition. Their motivation comes from the realization that given the size of modern software projects it is impractical to describe them with single models. Instead, they propose breaking the system into a set of models that offer complementary views which can be subsequently composed. Our work has a similar motivation, breaking a model that describes all posible members of a product line into features that can be later composed. Model management tools such as MOMENT [8] provide support for generic management operations and also rely on Maude.

Models and Software Product Lines. Work that couples FOSD and MDD into a framework called *Feature-Oriented Model Driven Development (FOMDD)* [4] has shown how to synthesize SPL products with MDD by first composing models that implement features which are subsequently transformed into executables. This work emphasizes the need for adequate support for algebraic model composition.

7 Conclusions and Future Work

The driving motivation of this work was to perform a preliminary assessment of the feasibility of using equational logic to implement FOSD concepts. Maude attracted our

attention for its native support of operator properties and its growing use for model management. Maude's equational logic appealed as a natural mechanism to realize the basic FOSD algebra and thus attempt to bridge the conceptual gap between FOSD concepts and their implementation. We found that the implementation of the kernel module required a significant design effort that later paid off as it facilitated the development of the class diagram module.

Performance was not an issue for the case study used to evaluate our implementation. However, this aspect needs further assessment with larger and more complex product lines. This assessment is part of our future work. Extensibility is another crucial issue to address. Work is underway to extend the algebra for state machines and sequence diagrams. We also plan to broaden our scope outside UML diagrams. We intend to study how our experience implementing the basic algebra could be leveraged for the development of other feature algebras. Our ultimate goal is to provide a platform in which FOSD concepts could be easily evaluated and experimented with.

Acknowledgments. We thank Sven Apel for the Java version of GPL case study.

References

1. Batory, D., Sarvela, J.N., Rauschmayer, A.: Scaling Step-Wise Refinement. IEEE TSE **30**(6) (2004)
2. Pohl, K., Bockle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer (2005)
3. Apel, S., Lengauer, C., Möller, B., Kästner, C.: An algebra for features and feature composition. In: AMAST. (2008)
4. Trujillo, S., Batory, D., Diaz, O.: Feature Oriented Model Driven Development: A Case Study for Portlets. In: ICSE. (2007)
5. Batory, D.S.: Using modern mathematics as an fosd modeling language. In Smaragdakis, Y., Siek, J.G., eds.: GPCE, ACM (2008) 35–44
6. Rivera, J.E., Vallecillo, A., Durán, F.: Maudeling: Herramienta de gestión de modelos usando maude. In: JISBD. (2007)
7. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.L.: All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic. Springer (2007)
8. Boronat, A., Carsí, J.A., Ramos, I., Letelier, P.: Formal model merging applied to class diagram integration. Electr. Notes Theor. Comput. Sci. **166** (2007)
9. Romero, J., Rivera, J., Duran, F., Vallecillo, A.: Formal and Tool Support for Model Driven Engineering with Maude. Journal of Object Technology **6**(9) (2007)
10. Rivera, J.E., Vallecillo, A.: Representing and operating with model differences. In Paige, R.F., Meyer, B., eds.: TOOLS (46). Volume 11 of Lecture Notes in Business Information Processing., Springer (2008) 141–160
11. Lopez-Herrejon, R.E., Batory, D.S.: A standard problem for evaluating product-line methodologies. In Bosch, J., ed.: GCSE. Volume 2186 of Lecture Notes in Computer Science., Springer (2001) 10–24
12. Lopez-Herrejon, R., Batory, D.: Modeling Features in Aspect-Based Product Lines with Use Case Slices: An Exploratory Case Study. In: Workshops and Symposia at MoDELS. (2006)
13. Herrmann, C., Krahn, H., Rumpe, B., Schindler, M., Volkel, S.: An Algebraic View on the Semantics of Model Composition. In: ECMDA-FA. (2007)

Revisión Sistemática de Métricas de Calidad para Líneas de Productos Software

Sonia Montagud, Silvia Abrahão

Grupo de Investigación ISSI, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera, s/n, 46022, Valencia, España
{smontagud, sabrahao}@dsic.upv.es

Abstract. En los últimos años se ha propuesto una cantidad considerable de métricas para evaluar características de calidad de líneas de productos software (LPS). Sin embargo, no existe ningún estudio que sintetice el conocimiento actual sobre ellas. Este artículo presenta una revisión sistemática sobre métricas de calidad para líneas de productos software propuestas por investigadores desde el año 1996. Las métricas han sido clasificadas teniendo en cuenta los procesos del ciclo de vida de las líneas de producto software y las características de calidad propuestas en la ISO/IEC SQuaRE. Se han obtenido un total de 104 métricas. Los resultados de la revisión indican que el 61% de las métricas miden atributos relacionados con la mantenibilidad. Además, el 68% de las métricas se utilizan durante la fase de diseño en la ingeniería del dominio y un 54% se aplican sobre la arquitectura de la línea de productos. Sin embargo, sólo un 1% de ellas han sido validadas empíricamente. Estos resultados son particularmente relevantes para la definición de un modelo de calidad para LPS.

Keywords: calidad, métricas, líneas de productos software, revisión sistemática

1 Introducción

La calidad es un factor crucial en la Ingeniería del Software. Si bien es importante para la construcción de productos individuales, cobra mayor relevancia en la ingeniería de Líneas de Productos Software (LPS), donde se pretende garantizar la calidad de todos los productos de una familia. Para la evaluación de la calidad, las métricas software son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento [6].

En los últimos años han aparecido numerosas métricas para evaluar la calidad en LPS. En [10] se comenta el uso de algunas métricas como un método para evaluar la calidad de las arquitecturas de líneas de productos, pero no las analizan en profundidad. Para averiguar si existe algún estudio que recoja y analice todas las métricas existentes para la evaluación de calidad en LPS, hemos realizado diferentes búsquedas en las bibliotecas digitales IEEEXplore, ACM Digital Library e INSPEC, no obteniendo ningún resultado. Con el propósito de conocer dicha información, este trabajo presenta una revisión sistemática de la literatura que recopila y analiza las métricas existentes para la evaluación de la calidad en LPS. Las revisiones

sistemáticas son útiles para identificar, evaluar, e interpretar una pregunta de investigación, un área de estudio, o un fenómeno de interés de forma objetiva [13].

Con respecto a otras revisiones sistemáticas relacionadas con este tema, en [14] hemos presentado una revisión sistemática donde se analizan todos los métodos y técnicas existentes para la evaluación de calidad en LPS, pero no se centra específicamente en la búsqueda y clasificación de métricas para LPS.

En lo referente al método de investigación, en la literatura podemos encontrar otros trabajos en los que se realiza una revisión de la literatura para descubrir y analizar las métricas existentes en otras áreas de la ingeniería del software ([7], [9]). Así, parece razonable el uso de una revisión sistemática como método de investigación.

La organización del artículo es la siguiente. En la Sección 2 se explica el protocolo de la revisión. La Sección 3 presenta los resultados obtenidos. La Sección 4 discute las posibles amenazas que puedan afectar a los resultados. Finalmente, en la Sección 5 se comentan las conclusiones y los trabajos futuros.

2 Método de Investigación

Una revisión sistemática es un método de investigación con el que obtener, evaluar e interpretar de forma fiable, rigurosa y metodológica toda la información que está relacionada con una pregunta de investigación o un área de interés particular. En este estudio, hemos seguido el método propuesto por Kitchenham [13], que consta de tres actividades principales: planificación, revisión y presentación de resultados.

2.1 Planificación de la revisión

2.1.1. Formulación de la pregunta de investigación: El objetivo de esta revisión es obtener y analizar las métricas que han sido utilizadas por los investigadores para evaluar la calidad de líneas de productos software y cómo han sido utilizadas.

2.1.2. Selección de Fuentes: Se ha intentado recoger las bibliotecas digitales de las organizaciones y editoriales más relevantes en Ingeniería del Software. La selección es la siguiente: *IEEEXplore*, *ACM Digital Library*, *Science Direct* e *INSPEC*. Además, se ha considerado las siguientes actas de congresos relacionados a: (i) líneas de productos software (SPLC, PFE y IWSAPF); (ii) calidad del software (ESEM, ISESE y METRICS); y (iii) ingeniería del software (ICSE y ECSA). Además, hemos añadido literatura gris con el fin de ampliar las fuentes del estudio.

El período de búsqueda comprende desde el año 1996 (año en el que se celebró el primer congreso dedicado específicamente a LPS) hasta principios del año 2009.

Tras refinar la cadena de búsqueda para optimizar los resultados, ésta fue: *(metric* OR measur*) AND (software) AND ("product line*" OR "product famil*")*.

La búsqueda se llevó a cabo utilizando el título y el resumen de los artículos.

2.1.3. Definición de criterios de Inclusión y Exclusión: Se ha incluido en la revisión artículos que presenten métricas para evaluar la calidad en LPS. Se han excluido los artículos que no estaban escritos en inglés, que no proponían métricas

directamente relacionadas con la calidad, que proponían métricas pero que no explicaban cómo medirlas o que definían atributos pero no sus métricas.

2.1.4. Estrategia de Extracción de Información: Una vez identificados y obtenidos los estudios primarios, la siguiente fase consistió en su revisión y en la extracción de la siguiente información:

1. **Característica de calidad evaluada**. Cada métrica se ha clasificado en función de la característica de calidad que pretende medir, según el modelo de calidad del estándar ISO/IEC SQuaRE [11]: *funcionalidad*, *fiabilidad*, *usabilidad*, *eficiencia*, *mantenibilidad* y *portabilidad*.
2. **Fase del ciclo de vida en el cual se aplica la métrica**. De acuerdo a [17], las fases son: Ingeniería del Dominio: (1) *Requisitos,* (2) *Diseño,* (3) *Realización,* (4) *Pruebas;* Ingeniería de la Aplicación: (5) *Requisitos*, (6) *Diseño,* (7) *Realización,* (8) *Pruebas*. Finalmente, incluimos la fase: (9) *Evolución*, sugerida por Bosch [5].
3. **Artefacto sobre el cual se realiza la evaluación**. Clasificamos las métricas, en función del artefacto evaluado, de la siguiente forma: (1) *Arquitectura Base*, (2) *Activos*, (3) *Arquitectura del Producto*, (4) *Producto final*.
4. **Procedimiento de validación de la métrica**. Existen dos tipos: *Validación teórica* o *Validación empírica*. Si carece de validación, se clasifica como *No validada.*
5. **Soporte para la medición**. *Manual* o *Automática*.

2.2 Conducción de la revisión

La búsqueda en las bibliotecas digitales se realizó el 10/03/2009 e identificó 442 potenciales publicaciones. Los criterios de inclusión y exclusión seleccionaron 16 artículos: 4 procedentes de la búsqueda manual, 9 de la búsqueda automática y 3 considerados literatura gris. Las publicaciones que aparecieron en más de una fuente se consideraron una vez con el siguiente orden de prioridad: (1) Actas de Congresos, (2) IEEEXplore, (3) ACM, (4) Science Direct e (5) Inspec. La Figura 1 lista los artículos seleccionados.

1. Abdelmoez, W. et al.: Error Propagation In Software Architectures. METRICS, 2004.
2. Ajila, S. A., Dumitrescu, R. T.: Experimental use of code delta, code churn, and rate of change to understand software product line evolution. JSS, 2007.
3. Alves de Oliveira Junior, E. et al.: A Metric Suite to Support Software Product Line Architecture Evaluation. CLEI, 2008.
4. Aldekoa, G. et al.: Quantifying Maintainability in Feature Oriented Product Lines. CSMR, 2008.
5. Dincel, E. et al.: Measuring Product Line Architectures. PFE, 2001.
6. Ganesan, D. et al.: Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG. SPLC, 2007.
7. Inoki, M., Fukazawa, Y.: Software Product Line Evolution Method Based on Kaizen Approach. SAC, 2007.
8. Johansson, E., Höst, R.: Tracking Degradation in Software Product Lines through Measurement of Design Rule Violations. SEKE, 2002.
9. Mišić, V. B.: Measuring the Coherence of Software Product Line Architectures. TR 06/03. Univer. of Manitova, Canada, 2006.
10. Needham, D., Jones, S.: A Software Fault Tree Metric. ICSM, 2006.
11. Rahman, A.: Metrics for the Structural Assessment of Product Line Architecture. Master Tesis on Software Engineering, Thesis no. MSE-2004:24. School of Engineering, Blekinge Institute of Technology, Sweden, 2004.
12. Siegmund, N. et al.: Measuring Non-functional Properties in Software Product Lines for Product Derivation. APSEC, 2008.
13. Van der Hoek, A. et al.: Using Services Utilization Metrics to Assess the Structure of Prod. Line Arch. METRICS, 2003.
14. Sun Her, J. et al.: A framework for evaluationg reusability of core asset in product line engineering. IST, 2007.
15. Yoshimura, K. et al.: Assessing Merge Potencial of Existing Engine Control Systems into a Product Line. SEAS, 2006.
16. Zhang, T. et al.: Some Metrics for Accesing Quality of Product Line Architecture. CSSE, 2008.

Figura 1. Artículos seleccionados.

3 Resultados

La revisión sistemática ha localizado 104 métricas que han sido clasificadas siguiendo los criterios descritos en la sección 3.1.4. Seguidamente, se presenta un análisis de los resultados para cada criterio:

1. **Característica de calidad evaluada**: Los resultados han mostrado que la mayoría de las métricas extraídas de los artículos pretenden medir atributos relacionados con la Mantenibilidad (ver Figura 2(a)). Por ejemplo, Aldekoa *et al.* [4] proponen una métrica para calcular el Índice de Mantenibilidad de una línea de productos; Alves de Oliveira *et al.* [3] proponen un conjunto de métricas para calcular la complejidad de la arquitectura de la línea de productos, obtenidas a partir de la herramienta SDMetrics; Ganesan et al. [8] proponen un conjunto de métricas para calcular el coste de la realización de pruebas en los componentes. Con relación a la eficiencia, la segunda característica con más métricas, se encuentran, entre otras, las métricas propuestas por Van der Hoek *et al*. [15] (la opcionalidad y la variabilidad) que calculan ratios de utilización de servicios provistos y requeridos y evalúan la arquitectura de una línea de productos, basándose en dos características específicas de las mismas. Sun Her et al. [16] evalúan la reusabilidad de los activos. En general, la mayoría de las métricas están relacionadas con la reusabilidad de componentes y la variabilidad.

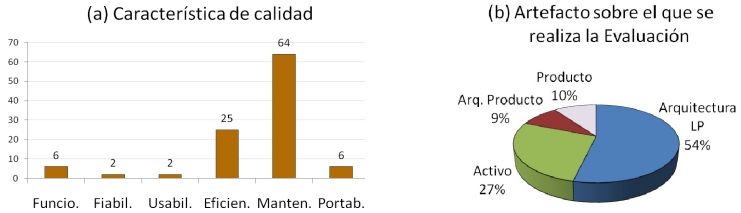


Figura 2. Resultados del criterio 1 (gráfica a) y criterio 3 (gráfica b).

2. **Fase del Ciclo de Vida en el que se aplica la Métrica**. Una gran cantidad de métricas (75) se aplican en la fase del diseño de la Ingeniería del Dominio. Esto es debido a que muchas de ellas están enfocadas a características concretas de las líneas de productos tales como la reusabilidad o la variabilidad, aspectos que se tienen muy presente en esta fase del ciclo de vida. La segunda fase más evaluada es la Evolución (14). Por ejemplo, Ajila *et al*. [2] identifican un conjunto de métricas (Tamaño de código en la línea de productos, Número de módulos, Código modificado, etc.) que pueden ser utilizadas para registrar los cambios en la línea de productos, con las cuales podrá observarse su evolución.

3. **Artefacto sobre el cual se realiza la Evaluación**. Como se puede observar en la Figura 2 (b), la mayoría de las métricas miden artefactos obtenidos en etapas tempranas del proceso de desarrollo de una línea de productos: el 54% de las métricas se aplican sobre la arquitectura de la línea de productos mientras que un 27% se aplican sobre los activos. En algunos casos, se utiliza más de un artefacto para realizar la medición. Por ejemplo, Johansson y Höst [12] hacen uso del código fuente, archivos y otros artefactos para comprobar las violaciones del diseño en el producto.

4. **Procedimiento de Validación de la Métrica**. Los resultados han mostrado que el 87% de las métricas carecen de cualquier tipo de validación. Sólo un 12% de ellas han sido validadas teóricamente. Éstas son las métricas propuestas por Sun Her *et al.* [16], las cuales han sido, por una parte, analizadas teóricamente mediante la aproximación propuesta por Kitchenham *et al*. [13], y por otra, aseguradas mediante seis criterios derivados del estándar IEEE Std 1061. Únicamente la métrica *"Error propagation"* propuesta por Abdelmoez *et al*. [1] ha sido validada empíricamente. La validación ha consistido en un estudio comparativo entre los resultados analíticos y los conseguidos experimentalmente. Sin embargo, ninguna métrica propuesta ha validado empíricamente la relación existente entre los valores obtenidos por la métrica y una característica de calidad externa.

5. **Soporte para la Evaluación**. El 80% de las métricas no disponen de una herramienta que de soporte a su medición. Como ejemplos de herramientas para la evaluación del restante 20% de las métricas, se encuentran la combinación de la herramienta Crystal Ball con la simulación Monte-Carlo para analizar los resultados de las métricas propuestas por [8] en un documento Excel; o Zhang *et al*. [18] que definen sus métricas sobre una especificación formal de la arquitectura de línea de productos con la especificación vADL, la cual permite hacer un análisis automático.

4 Amenazas a la Validez

Las posibles limitaciones de este estudio están relacionadas con el sesgo en la selección de publicaciones, el posible uso de una cadena de búsqueda incompleta y la inexactitud en la extracción de datos o errores en la clasificación. Esperamos haber disminuido la primera amenaza a través de la exploración manual de actas de congresos y estudios considerados *literatura gris*. Al seleccionar las publicaciones, tratamos de elegir las fuentes en las que suelen publicarse artículos relacionados con métricas y LPS. Con respecto a la segunda, procuramos recoger las cadenas que son representativas de la pregunta de investigación, refinamos la búsqueda varias veces, consideramos sinónimos e incluimos los lexemas de las palabras. Finalmente, intentamos suavizar la tercera amenaza realizando la clasificación con dos revisores.

5 Conclusiones y Trabajo Futuro

Se ha presentado una revisión sistemática de la literatura con el propósito de localizar y obtener información relevante sobre las métricas utilizadas por los investigadores en los últimos años para la evaluación de la calidad en líneas de productos software.

Los resultados son interesantes para conocer el estado actual de las métricas que pueden ser utilizadas por los investigadores o profesionales de la industria para evaluar la calidad en LPS (la mayoría evalúan la mantenibilidad (61%) y la eficiencia (24%)), o en qué fases son comúnmente utilizadas (el 68% en la fase de diseño de la Ing. del Dominio). Además, el estudio permite detectar carencias para direccionar los esfuerzos de investigación. Así, sería de interés: definir métricas que evalúen otras

características de calidad; validar las métricas existentes con estudios empíricos, definir métricas que evalúen la composicionalidad (combinación de la arquitectura con otros activos) para permitir decidir que diseño es mejor; cubrir la evaluación en todo el ciclo de vida; o desarrollar herramientas que faciliten la medición.

Como trabajo futuro, estamos trabajando en la definición de un modelo de calidad para LPS a partir de la descomposición del modelo de calidad propuesto en el estándar SQuaRE [11]. En él integraremos las métricas obtenidas en el presente trabajo y añadiremos las necesarias para completar el modelo.

Referencias

1. Abdelmoez, W., Nassar, D.M., Shereschevsky, M., Gradetsky, N., Gunnalan, R., Ammar, H.H., Yu, B., Mili, A.: Error Propagation in Software Architectures. In 10th International Symposium on Software Metrics (METRICS), Chicago, Illinois, USA, 2004.
2. Ajila, S. A., Dumitrescu, R. T.: Experimental use of code delta, code churn, and rate of change to understand software product line evolution. JSS, 80, pp.74-91, 2007.
3. Alves de Oliveira Junior, E., Gimenes, I. M. S., Maldonado, J. C.: A Metric Suite to Support Software Product Line Architecture Evaluation. In XXXIV Conferencia Latinamericana de Informática (CLEI), Santa Fé, Argentina, pp.489-498, 2008.
4. Aldekoa, G., Trujillo, S., Sagardui, G., Díaz, O.: Quantifying Maintainability in Feature Oriented Product Lines. In Eur. Conf. on SW Maintenance and Reengineering, 2008.
5. Bosch, J.: Design and use of software architectures: adopting and evolving a product line approach". ACM Press/Addison-Wesley Publishing Co., USA, 2000.
6. Briand, L.C., Differing, C.M., Rombach, D.: Practical Guidelines for Measurement-Based Process Improvement. Software Process-Improvement and Practice 2, pp.253-280, 1996.
7. Calero, C., Ruiz, J., Piattini, M.: Classifying web metrics using the Web Quality Model. Online Information Review, OIR - 29, 3 Emerald Literari. United Kingdom.
8. Ganesan, D., Knodel, J., Kolb, R., Haury, U., Meier, G.: Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG. In 11th International Software Product Line Conference, Kyoto, Japan, pp.74-83, September 2007.
9. Gómez, O., Oktaba, H., Piattini, M., García, F.: A Systematic Review Measurement in Software Engineering: State-of-the-Art in Measures. In First International Conference on Software and Data Technologies (ICSOFT), Setúbal, Portugal, pp.11-14. September, 2006.
10. Etxeberria, L., Sagarui, G., Belategi, L.: Quality aware software product line engineering. Journal of the Brazilian Computer Society, vol.14, no.1, Campinas Mar, 2008.
11. ISO/IEC 25000:2005. SW Eng. SW product Quality Requirements and Eval. (SQuaRE).
12. Johansson, E., Höst, R.: Tracking Degradation in Software Product Lines through Measurement of Design Rule Violations. Conf. SW Eng. and Knowledge Eng., Italy, 2002.
13. Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering. Version 2.3, EBSE Technical Report, Keele University, UK.
14. Montagud S., Abrahão, S.: Gathering Current Knowledge about Quality Evaluation in Software Product Lines. In International Software Product Lines Conferences, USA, 2009.
15. Van der Hoek, A., Dincel, E., Medidović, N.: Using Services Utilization Metrics to Assess the Structure of Product Line Architectures. In 9th Int. Software Metrics Symposium, 2003.
16. Sun Her, J., Hyeok Kim, J., Hun Oh, S., Yul Rhew, S., Dong Kim, S.: A framework for evaluationg reusability of core asset in product line engineering. IST 49, pp. 740-760, 2007.
17. Van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action. 2007.
18. Zhang, T., Deng, L., Wu, J., Zhou, Q., Ma, C.: Some Metrics for Accesing Quality of Product Line Architecture. Int. Conf. Computer Science and Software Engineering, 2008.

FMT: Una Herramienta de Modelado y Configuración de Líneas de Productos Software para MS Visual Studio*

Rubén Fernández, Miguel A. Laguna, Jesús Requejo, Nuria Serrano

Departamento de Informática, Universidad de Valladolid.
{luifern, mlaguna}@infor.uva.es, {jesus.requejo, nuriaserrano}@gmail.com

Resumen. Los modelos de características son instrumentos básicos para modelar, analizar y configurar la variabilidad de una línea de productos software. Todas estas tareas son complejas por lo que es necesario disponer de herramientas que satisfagan una serie de requisitos mínimos: generación automática de modelos, configuración de productos finales, integración en el entorno de desarrollo, etc. Por esa razón, hemos desarrollado una herramienta para realizar estas tareas, definiendo un lenguaje específico de dominio e implementando el mismo mediante las herramientas proporcionadas por Microsoft para Visual Studio. De esta forma hemos logrado integrar la herramienta en la misma plataforma utilizada para desarrollar la línea de productos, con lo que todas las fases de desarrollo comparten el mismo entorno.

1 Introducción

Uno de los problemas en el desarrollo de una línea de productos software (LPS) es la representación y gestión de la parte común y variable de la misma. La forma habitual de definir ambos aspectos es mediante modelos de características o features que permiten seleccionar la configuración de cada aplicación concreta dentro de la LPS. En la mayoría de LPS esas tareas son demasiado complejas para realizarlas manualmente. Por otro lado, en [3] hemos mostrado cómo se puede generar automáticamente la arquitectura básica de una línea de productos mediante transformaciones de los diagramas de características. La configuración de paquetes se basa en el mecanismo de *package merge* de UML 2 [4] y su implementación mediante clases parciales de C#. Hasta el momento para realizar todas estas tareas el grupo GIRO ha utilizado una combinación de herramientas: el *Feature Model Plug-in para Eclipse* de Czarnecki [1], y MS Visual Studio. La conexión entre ellas se realiza de manera manual utilizando ficheros XML y transformaciones con hojas de estilo XSLT. Los modelos de características se pueden convertir en la estructura de paquetes de una línea de productos, obtenida como un fichero XMI, que se puede importar con una herramienta CASE [3]. Actualmente existen diferentes herramientas libres que permiten la gestión de líneas de productos, como por ejemplo el mencionado *plug-in fmp* o Moskitt[1]. Todas ellas permiten la definición de modelos de

* trabajo financiado por la Junta de Castilla y León (VA018A07) y el MICIINN (TIN2008-05675)

[1] http://www.moskitt.org/eng/

características así como su configuración y gestión de la variabilidad. Sin embargo ninguna de ellas proporciona la funcionalidad que deseamos en una herramienta de estas características, como transformación automática de modelos, integración en el entorno de desarrollo que utilizamos o generación de las aplicaciones finales de la línea de productos.

Por esto, hemos desarrollado una herramienta, Feature Modeling Tool[2] (FMT), que nos permita realizar diagramas de características, así como gestionar la variabilidad, realizar configuraciones de productos específicos y además integrar todo ello en MS Visual Studio. Nuestra motivación es disponer de una herramienta que integre todas las fases del proceso en el mismo entorno de desarrollo y realice de manera automática las tareas que hasta el momento se hacen exportando e importado los distintos modelos de manera manual. En el resto del documento se describe la herramienta con más detalle y se muestra un ejemplo de uso.

2 Descripción de la Herramienta FMT

Para definir los modelos de características que crearemos con FMT hemos seleccionado el meta-modelo que utiliza Czarneki [1] en el *plug-in* para Eclipse. Hemos elegido este meta-modelo porque es uno de los más utilizados y para facilitar la compatibilidad con *fmp*, permitiendo la importación/exportación de modelos.

Para desarrollar FMT y poder integrarla en el mismo entorno de desarrollo que utilizaremos para desarrollar los productos, hemos utilizado DSL Tools [2]. DSL Tools es una herramienta incluida en el SDK de MS Visual Studio que permite definir lenguajes de dominio específico y crear herramientas de modelado para esos lenguajes. Entre las ventajas que aporta FMT podemos destacar la representación gráfica de los modelos de características incluyendo restricciones tipo *mutex/require* y sus posibilidades de automatización: transformación de modelos y generación de las aplicaciones finales de la LPS. La Figura 1 muestra la interfaz de FMT.

La ventana principal se divide en tres secciones. La ventana central es el panel de diseño, donde podemos crear los diagramas de características. La visión gráfica se complementa con el explorador en árbol, idéntico al formato utilizado por *fmp* (panel inferior derecho). La sección de la izquierda es el panel de configuración. Desde este panel podemos seleccionar aquellas características que queremos que formen parte del producto específico a desarrollar en cada caso. La propia herramienta se encarga de comprobar las multiplicidades y otras restricciones como las relaciones *require* y *mutex*. Por último, en la tercera sección de la ventana localizada en la parte derecha tenemos la configuración de la LPS como un proyecto de desarrollo (una solución de Visual Studio). En esta ventana podemos ver la transformación en el modelo de paquetes para nuestra configuración del diagrama de características. Si marcamos una característica en el panel de configuración su paquete correspondiente será añadido a la solución.

Otra función de la herramienta es la importación/exportación de modelos. Podemos importar un modelo creado previamente con el *plug-in fmp*. También

[2] disponible en http://giro.infor.uva.es/FeatureTool.html

podemos exportar nuestro modelo de características a un modelo UML (en formato XMI). Para ello la herramienta realiza la transformación propuesta en [3], utilizando las facilidades de manejo de ficheros XML proporcionados por la plataforma. Otra de las funciones permite la exportación en el formato de la herramienta FaMa [5] que realiza la validación de los modelos de características.

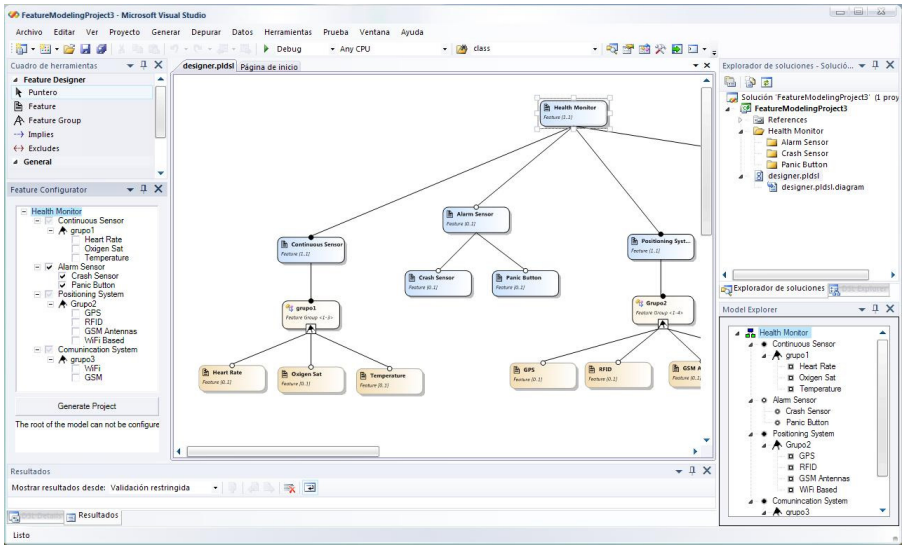


Fig. 1 Interfaz de FMT

3 Ejemplo de Uso

Para mostrar el funcionamiento de la herramienta, la Figura 2a muestra el flujo de trabajo a seguir en el desarrollo de los activos de la LPS. Inicialmente se crea el modelo de características. Podemos hacer esto de dos maneras: bien creándolo manualmente en la ventana de diseño o bien importando un modelo ya existente desde el *plug-in fmp*. Ahora podemos hacer cualquier modificación que creamos conveniente sobre este modelo. Podemos añadir restricciones gráficas que el *plug-in fmp* no permite, como relaciones *require* y *mutex* o asignar un tipo a las características hoja. Una vez finalizado el diagrama de características de nuestra línea de productos, el siguiente paso es crear la solución de Visual Studio para comenzar el desarrollo. FMT crea automáticamente la solución generando los paquetes correspondientes a la transformación del modelo de características. Si seleccionamos la opción "*export model*", la herramienta realiza la transformación del modelo de características a un modelo de paquetes UML.

El siguiente paso consiste en completar esos paquetes. Una vez que los paquetes de la LPS están completos podemos configurar aquellos productos finales que deseemos. Basta con seleccionar en el panel de configuración aquellas características que queremos que formen parte del producto y Visual Studio seleccionará los paquetes necesarios y compilará la solución (Figura 2b).

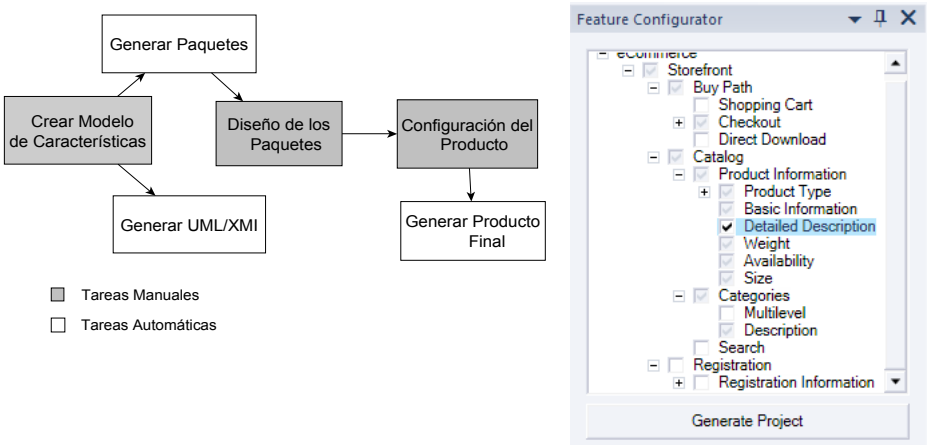


Fig. 2 Flujo de trabajo para el desarrollo de una LPS (a) y herramienta de configuración (b)

4 Conclusiones

Como resumen final, hemos desarrollado una herramienta que no solo nos permite gestionar y configurar diagramas de características, sino que también realiza de forma automática las transformaciones entre los diferentes modelos, creando la arquitectura básica de la línea de producto, automatizando la mayoría de las tareas del proceso de desarrollo e integrando toda la funcionalidad en la misma herramienta que será utilizada posteriormente para la generación de los productos finales de la LPS. Se encuentra en fase de integración la generación automática de interfaces de usuario a partir de la combinación de archivos XML parciales.

Referencias

1. M. Antkiewicz and K. Czarnecki. “Feature Plugin: Feature modeling plug-in for Eclipse.” In OOPSLA’04 Eclipse Technology eXchange (ETX) Workshop, 2004.
2. Steve Cook, Gareth Jones, Stuart Kent, Alan Cameron Wills. “Domain-Specific Development with Visual Studio DSL Tools (Microsoft .NET Development Series)” Junio 2007. Addison-Wesley Professional
3. Miguel A. Laguna, Bruno González-Baixauli.”Variabilidad, Trazabilidad y Líneas de Productos: una Propuesta basada en UML y Clases Parciales.” JISBD 2007, Zaragoza, Spain - sep 2007
4. Object Management OMG. “Unified modeling language specification version 2.0: Infrastructure.” Technical Report ptc/03-09-15, OMG, 2003.
5. Pablo Trinidad, David Benavides, Antonio Ruiz Cortés, Sergio Segura, Alberto Jiménez. FAMA Framework. SPLC 2008: 359

Moskitt FM and FAMA FW: Taking feature models to the next level

Carlos Cetina[1], Pablo Trinidad[2], Vicente Pelechano[1], Antonio Ruiz-Cortés[2], David Benavides[2]

[1] Centro de Investigación en Métodos de Producción de Software
Universidad Politécnica de Valencia, Camino de Vera s/n, E-46022, Spain
{ccetina, pele} at dsic dot upv dot es
[2] Universidad de Sevilla, Spain
{ptrinidad, aruiz} at us dot es

Abstract. Feature modeling is a key technique to model commonalities and variabilities within a Software Product Line (SPL). Tools to realise the benefits of using feature models are needed to make true the benefits promised by SPL approaches. This paper introduces the integration of MOSKitt Feature Modeller (MFM) and FaMa Framework (FaMa FW) to support feature modelling and analysis in the context of SPLs. On the one hand, MFM provides advance capabilities for feature model edition. On the other hand, FAMA FW automates their analysis. The result is an extensible tool that should help SPL developers on leveraging feature models.

1 Moskitt Feature Modeler

Models are valuable documentation assets since they capture relevant information of a system. However, they can play a more relevant role in the development process of a system. MDE proposes their use as central assets for system development. MDE tools enable the automatic manipulation of models to obtain new assets (other models, software systems, documents, etc.). But to build machine-processable models, they must be given a precise semantics.

Moskitt[3] is an open source modular modelling set of tools to support the development process defined by the Infrastructures and Transport Ministry of the Generalitat Valenciana in Spain. The tool is based on Eclipse and provides support for model management (graphical edition, persistence, cooperative work, etc.) It also allows to establish relationships between models (generation of models and production of traces, dependencies, consistency checks, etc.) and the production of new assets taking the models as input.

Moskitt supports different kind of models as left of Fig 1 shows. UML models are supported. These diagrams are based on the full UML 2.0 metamodel defined by the OMG. In addition to UML, other models are supported to cover particular needs. Moskitt provides support for the modeling of relational database schemata (that can

[3] http://www.moskitt.org/

Actas JISBD 2009, pp.285-288, ISBN: 978-84-692-4211-7

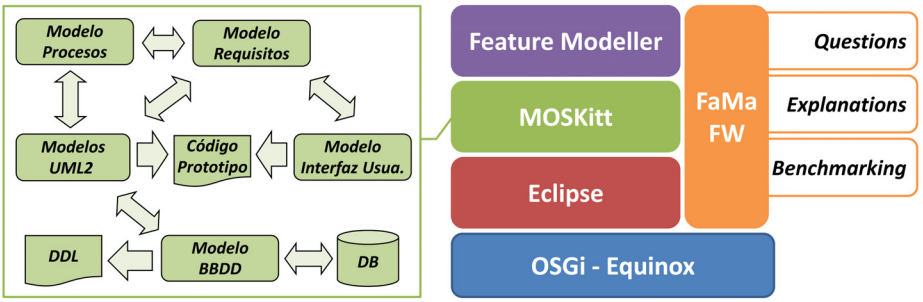


Fig. 1. Overview of the MFM-FaMa Platform.

be derived from UML models). It also provides a Requirements Editor for the generation and maintenance of a requirements catalog and support for traceability between requirements and the rest of models. User Interface and Business Process modeling is also supported by the tool.

Feature Modeling has an important role in the software industry. Although an isolated modeling tool for feature models can be interesting by its own, the integration with a MDE method and tools for software development provides an added-value. Moskit Feature Modeler (MFM) is the open source feature model editor of Moskitt. MFM shares several technologies with Moskitt editors and takes advantage of some infrastructure components defined by Moskitt (such as a manager for model-to-model transformations). Thanks to the support for defining relationships among models, feature models can be used in conjunction with the existing editors (without modification) to enable their use for Product Line Engineering.

As a result, MFM has been integrated in the Moskitt tool set (see Fig 2), providing edition capabilities for Feature Models that can be used along the software developent process. Therefore, a new opportunity to the MDE community for the development of new tools and extensions to operate on those models are provided.

The main characteristicts that MFM presents in feature model manipulation include:

- **Notation Flexibility**. The graphical notation used to represent the elements in a model can be changed dynamically.
- **Automatic Tree Layout**. The MFM can reorganize the elements of a model in a graphical tree, in which elements at the bottom are the leaves of the tree, the element on the top is the root of the tree and the rest of the elements are ordered by levels.
- **Features Explosion**. Any feature can be exploded in another feature model. In addition, the graphical representation of exploded features changes to provide information about its submodel population.
- **Interoperability with FMP**. MFM models can be transformed into Antkiewicz and Czarnecki's Feature Modeling Plug-in tool models.

At MFM website[4] the source and binary distribution of Moskitt Feature Modeler are available. Furthermore, some screencasts are also available.

[4] http://www.pros.upv.es/labs/projects/mfm

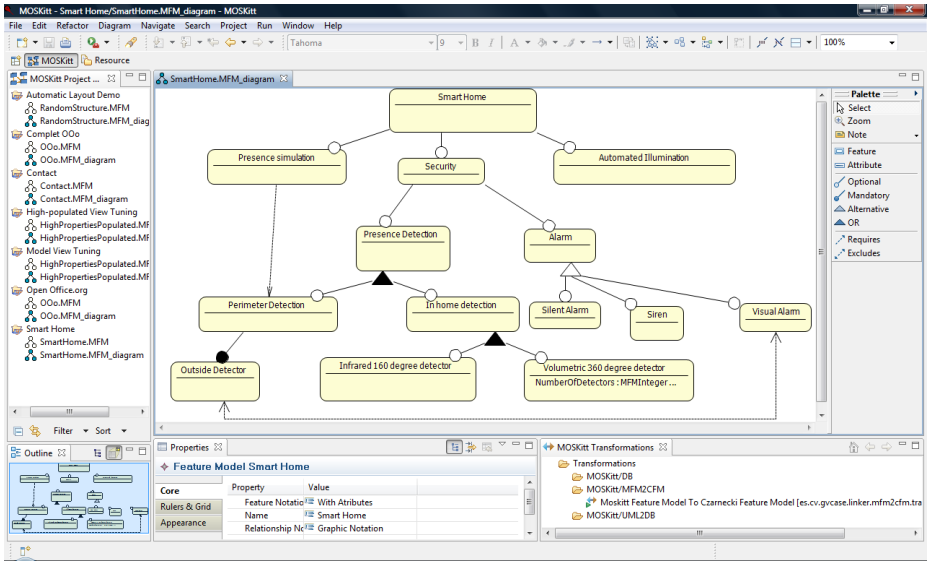


Fig. 2. MFM Screenshot.

2 FAMA Framework

The analysis of feature models is an important activity to support decisions in software product lines (SPL) development. Analysis operations such as counting the number of potential products, filtering them following a criteria, searching for an optimal product in development cost or time, detecting and explaining errors in a feature model,... are very important in all the activities where feature models are used [1–3]. As feature models may be quite large and contain complex relationships, analysing them by hand is not a suitable activity, so an automated error is needed. To realise such automated analysis of feature models (AAFM), we map feature models into different declarative paradigms such as constraint satisfaction problems (CSP), satisfiability problems(SAT), binary decision diagrams (BDD) among others. Each paradigm has its pros and cons, performing better or worse than other paradigms depending on the kind of operation or the feature model to analyse.

FAMA FW [5] [4] arises as the reflex of our experience in AAFM into a flexible tool. FAMA FW is a Java framework from the point of view of AAFM researchers that want to integrate their theoretical results taking advantage of the services it provides such as metamodel-independence, file storing, random generation of feature models and end-user applications integration. Moreover, FAMA FW is a library from the point of view of consumers since it offers a stable facade that allows interacting with the tool independently from the available metamodels, reasoners or questions.

[5] http://www.isa.us.es/fama

FAMA FW is a SPL itself, as it allows the customer to decide which extensions to incorporate in a final distribution of FAMA FW and even extend it in the future, needing no kind of adaptation. Among its extensions we find:

- Reasoners: CSP (JaCoP and Choco), SAT (SAT4j) and BDD (JavaBDD).
- Metamodels: FAMA Metamodel, Moskitt FM Metamodel. Stores and retrieves feature models in XML and X3D formats.
- Questions: number of products, list of products, error detection, error explanation, valid configurations, filtering, etc.
- Reasoner Selector: different criteria to select the reasoner with a best performance in answering a specific question.

With the bearing point of a further OSGi integration, FAMA FW was built following an OSGi-inspired architecture where every extension is an auto-contained Java Jar file and dependencies among them are contained in a configuration file.

3 MFM and FAMA FW Integration

FeAture Model Analyser (FAMA) was presented in a previous edition of JISBD as a powerful tool for the AAFM but its Eclipse user interface was coupled to a metamodel that hindered a natural edition. Due to the experience Moskitt gave in developing Eclipse plugins, the UPV team offered to develop an intuitive feature model editor that could integrate with FAMA. USE team would invest their efforts in refactoring FAMA to produce a flexible AAFM platform in first term, and an OSGi-compliant platform in second term. FAMA FW is the result of such refactoring and was presented in last year edition of JISBD. This year we present the final result of both tools integration as a result of a new OSGi-compliant version of FAMA FW and the release of Moskitt Feature Modeler.

References

1. D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, 3520:491–503, 2005.
2. D. Benavides, A. Ruiz-Cortés, P. Trinidad, and S. Segura. A survey on the automated analyses of feture models. In *Jornadas de Ingeniera del Software y Bases de Datos (JISBD)*, 2006.
3. P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software*, 81(6):883–896, 2008.
4. P. Trinidad, D. Benavides, A. Ruiz-Cortés, S. Segura, and A.Jimenez. Fama framework. In *12th Software Product Lines Conference (SPLC)*, 2008.

Parte IX

Sesión 8. Ontologías, Web semántica

WEAPON: Modelo de Workflow con Ontologías para Procesos Administrativos

Álvaro E. Prieto, Adolfo Lozano-Tello

Universidad de Extremadura
Avda. de la Universidad s/n, 10071, Cáceres
{aeprieto,alozano}@unex.es

Resumen. Los procesos de gestión administrativa son un tipo de proceso de negocio caracterizados por la participación de diferentes usuarios involucrados en diferentes actividades del proceso, añadiendo, modificando o confirmando información contenida en elementos de datos perfectamente definidos y estructurados hasta la finalización del proceso. Estos procesos suelen ser muy reutilizables, ya que el mismo patrón de gestión suele usarse para resolver procesos similares dentro de una institución, adaptando en algunos casos el orden de ejecución de las actividades, los usuarios responsables de realizarlas o alguna de las estructuras de datos que manejan. Por este motivo, es muy adecuado que sean definidos en sistemas de gestión de workflow que faciliten la adaptación a dichos cambios. La representación en ontologías de los procesos de workflow y de los datos que manejan pueden facilitar las tareas de reutilización y adaptación de estos procesos de negocio. En este trabajo se presenta WEAPON, un modelo de diseño y gestión de workflow para procesos administrativos basado en ontologías, que es asistido por herramientas software.

Palabras clave: procesos de negocio, gestión administrativa, workflows, ontologías, WEAPON

1 Introducción

En los últimos años, los Sistemas de Gestión de Workflow (Workflow Management System, WfMS en el resto del texto) están ganando popularidad gracias a informes [1] que sitúan la Gestión de Procesos de Negocio (Business Process Management, BPM) como una prioridad para las empresas. La Workflow Management Coalition (WfMC[1]), que trata de estandarizar estos sistemas para facilitar su implantación, define un workflow como "la automatización de un proceso de negocio, en todo o en parte, durante el cual se transmite documentación, información o tareas de un participante a otro para llevar a cabo una acción, de acuerdo a una serie de reglas" [2]. A partir de esta definición se pude decir que un WfMS es capaz de interpretar la definición del workflow y crear y gestionar su ejecución.

[1] http://www.wfmc.org

La clasificación más conocida de los WfMS es la que propone McCready [3] distinguiendo entre procesos ad hoc, administrativos y de producción. La de Georgakopoulos [4] los clasifica como orientados a humano u orientados a sistemas y la realizada por Weske [5] hace una división entre mono-aplicación o multi-aplicación. Atendiendo a estas clasificaciones se puede identificar un tipo de WfMS orientado a la gestión administrativa, para ser utilizado por humanos y mono-aplicación. Este tipo de proceso no suele requerir de un WfMS complejo con características avanzadas tales como coordinación con otras aplicaciones externas o diferentes posibilidades en el control del flujo del proceso o en sus eventos. Más bien, el WfMS ideal para este tipo de procesos sería aquel dónde las actividades que componen el proceso, los elementos de datos que manejan dichas actividades y los usuarios involucrados en su realización están identificados y clasificados sin ambigüedades de manera que este tipo de WfMS sea fácilmente adaptable a los cambios que regularmente pueden ocurrir en este tipo de procesos, bien sea por cambios en las normativas que los rigen o bien por tratar de reutilizarlos en organismos o empresas similares.

Este trabajo se centra en este tipo de proceso de negocio, que generalmente se utiliza en el ámbito legal o de gestión administrativa, caracterizado por ser iniciados por un usuario y que debe ser atendido o evaluado por otros distintos siguiendo un protocolo perfectamente definido en cuanto a las estructuras de datos a transmitir, tiempos de realización y usuarios involucrados en cada actividad. En general son procesos regidos por leyes, normativas de actuación o protocolos de ejecución de instituciones públicas o empresas de gran tamaño. Por poner algunos ejemplos, podrían incluirse en este tipo los procesos de gestión de presentación de ofertas a concurso público, peticiones de préstamos bancarios, o un simple procedimiento de solicitud de vacaciones.

El uso de ontologías como base para la gestión de este tipo de procesos puede ser muy útil por sus características de representación precisa y completa de los términos, su facilidad para integrar esquemas de datos similares y su disposición de reutilización y adaptación con un mínimo esfuerzo. Con este objetivo, en este trabajo se presenta un modelo de WfMS que permite definir y gestionar los procesos de gestión administrativa utilizando ontologías. Este modelo, denominado WEAPON (Workflow Engine for Administrative Processes based on ONtologies), nace con el propósito de favorecer la identificación precisa de los elementos de datos tratados así como de las actividades que componen los procesos y los usuarios involucrados, además de facilitar los procesos de adaptación de modelos de gestión administrativa existentes a otros similares.

Este trabajo está estructurado como sigue: en la sección 2 se identifican trabajos que usan ontologías en WfMS y en la sección 3 se detalla el modelo propuesto, compuesto por las ontologías y el sistema de workflow.

2 Uso de ontologías en WfMS

En los últimos años se ha extendido el uso de ontologías para representar conocimiento en cualquier dominio, tanto en sistemas basados en conocimiento y razonamiento como en sistemas de información tradicionales [6]. Esto es debido a que

están diseñadas con el objetivo de que su conocimiento sea fácilmente reutilizable y compartido por comunidades y usuarios de un mismo dominio. Además, la aceptación de OWL[2] (Ontology Web Language) como lenguaje de facto para su representación y el desarrollo de herramientas como Protégé[3] para su construcción ha favorecido su amplio uso en muchos campos, especialmente en la Web Semántica.

Aplicadas a la definición de procesos de negocio, las ontologías proporcionan una terminología completa, precisa y compartida sobre un dominio particular que facilita la integración y que será fácilmente reutilizable por la misma u otra organización. Estas ventajas proporcionan un ahorro considerable de tiempo y esfuerzo en las tareas de definición de procesos y datos, o en los métodos de mezcla cuando existen representaciones similares del mismo dominio.

En concreto, la aplicación de ontologías a los WfMS se ha utilizado con anterioridad en propuestas como la de Vieira et al. [7] que es, posiblemente, el primer trabajo integrando ambos campos y donde proponen una solución para flexibilizar la ejecución de los workflows. También es interesante el trabajo de Gasevic et al. [8] donde proporcionan una ontología de redes de Petri. En el trabajo de Haller et al. [9] presentan una ontología de multi meta-modelo de proceso (m3po), que relaciona modelos de workflow y de coreografía. Por último, Abramowicz et al. [10] han construido la ontología sBPMN que añade semántica a la notación BPMN (Business Process Modeling Notation) [11]. Además de las reseñadas también podemos citar las recogidas en [12,13,14,15,16] como ejemplos de unión de ambos campos.

A diferencia de las propuestas anteriores en este trabajo se presenta un modelo que se centra en los procesos de gestión administrativa utilizando ontologías para representarlos junto con sus actividades, datos relevantes y usuarios involucrados. Las ventajas que puede aportar con respecto a utilizar modelos y lenguajes de representación de workflows consolidados [17,18,19,20] son:

- Los usuarios, siguiendo metodologías de desarrollo de ontologías, pueden obtener definiciones de proceso completas, precisas y consensuadas.
- Los datos relevantes de un proceso o los usuarios que participan en él pueden ser intercambiados sin la necesidad de llevar a cabo modificaciones en las definiciones de los datos que manejan las actividades, ni en la definición de los procesos.
- Las definiciones de los workflows, representadas en ontologías, son más fácilmente reutilizables. Aunque dicho proceso de reutilización puede suponer un esfuerzo en cuanto a los procesos de búsqueda, selección, y en algunos casos, en la adaptación de las ontologías al nuevo sistema. Estos factores son discutidos en detalle en [21].

Algunas de estas ventajas también son descritas en Dang et al. [22], donde proponen el uso de ontologías como base para workflows en el dominio médico. Aunque su enfoque se restringe al uso de una ontología completa y cerrada sobre workflows del ámbito hospitalario. Nuestro modelo, en cambio, provee una ontología genérica para procesos administrativos y proporciona los métodos y herramientas necesarios para reutilizar, en parte o completamente, ontologías apropiadas de los workflows del dominio que se está representando.

2 http://www.w3.org/2004/OWL/

3 http://protege.stanford.edu/

3 Modelo de Workflow Basado en Ontologías usado en Procesos de Gestión Administrativa

El modelo presentado en este trabajo, denominado WEAPON (Workflow Engine for Administrative Processes based on ONtologies), propone el uso de ontologías para definir y gestionar procesos administrativos que se caracterizan por ser iniciados por un usuario y que deben ser atendidos o evaluados por otros distintos, siguiendo un protocolo perfectamente definido en cuanto al orden de las actividades a realizar, las estructuras de datos que manejan dichas actividades, los usuarios responsables de realizarlas y los tiempos disponibles para llevarlas a cabo.

Básicamente, el modelo propone cómo un usuario supervisor debe definir, por un lado, la taxonomía de datos relevantes al dominio y la taxonomía de usuarios que pueden participar en el workflow y, por otro, las actividades que componen el proceso junto a la identificación de los usuarios que pueden realizarlas y los datos que son gestionados en cada actividad. Esto asegura que los procesos están bien definidos y son más reutilizables y, además, las clases de los datos relevantes involucrados y las clases de los participantes del workflow pueden ser modificadas sin cambiar la representación del workflow del proceso. Como ventaja adicional debe destacarse que las taxonomías de clases e instancias que pueden ser necesarias como datos relevantes en el proceso administrativo, pueden estar definidas previamente en ontologías de la propia organización o pueden ser reutilizadas desde repositorios externos.

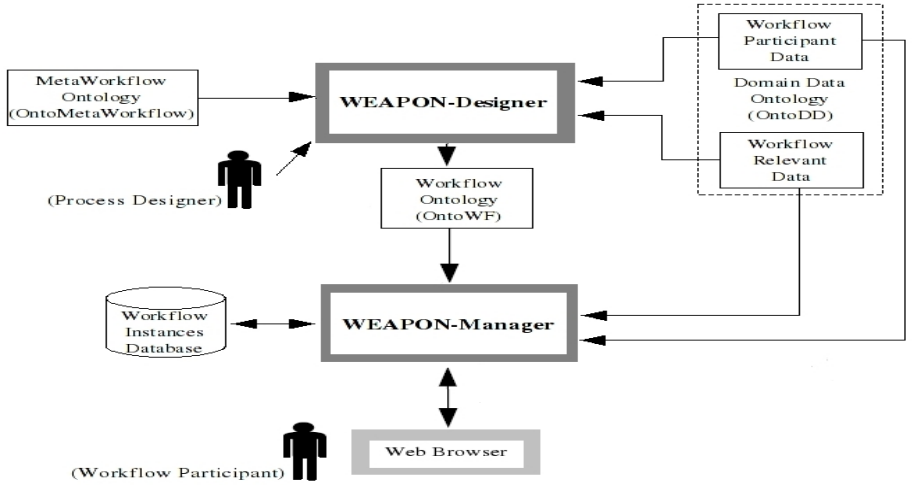


Fig. 1. Visión general del modelo WEAPON

Usando un ejemplo para explicar el modelo, si un proceso trata sobre la petición de un préstamo al banco por parte de un cliente, se necesitan, entre otros datos relevantes, los tipos de crédito que concede dicho banco o la catalogación interna que hace el banco de los solicitantes de préstamo. El diseñador del workflow podría reutilizar una ontología existente de dicho dominio sobre tipos de crédito y condiciones y catalogaciones de clientes con el consiguiente ahorro de tiempo. Pero

se pueden comprender las ventajas añadidas si suponemos que en este ejemplo es el Banco Central del país el que fija las condiciones que deben de seguir los procesos de solicitud de crédito por parte de los clientes, sería suficiente con tener la definición del workflow propuesta por el Banco Central y simplemente, adaptarla a cada banco en particular, cambiando las taxonomías de datos relevantes (tipos de crédito, condiciones, etc.) y los participantes (clientes del banco, empleados encargados de revisar las solicitudes, etc.).

El modelo presenta un conjunto de componentes relacionados que conforman su arquitectura (figura 1). Como se describe a continuación, nuestra propuesta proporciona una ontología como base para la representación de workflows, junto con los métodos (y sus respectivas herramientas software) para identificar y explotar los workflows del proceso administrativo. Los elementos que constituyen el modelo son:
1) una ontología para la definición genérica de workflows (OntoMetaWorkflow).
2) una ontología de de datos del dominio (OntoDD) a partir de OntoMetaWorkflow.
3) una ontología del workflow del proceso administrativo (OntoWF) a partir de OntoMetaWorkflow y OntoDD.
4) la herramienta WEAPON Designer que permite diseñar el proceso administrativo de forma gráfica y almacenarlo en la ontología OntoWF.
5) la herramienta WEAPON Manager que permite generar la aplicación web que actuará como WfMS del workflow modelado.

Hay que destacar que, dado que las actividades de este tipo de procesos están muy ligadas a los datos que manejan, se han utilizado en este modelo tanto ideas de los WfMS tradicionales como de alguna de las aportadas en la propuesta Case Handling [23]. Case Handling se propuso partiendo de la idea de que la gestión de muchos procesos no sólo debe ser en función de las actividades si no también en torno a los datos que manejan dichas actividades. Además, proponen el uso de formularios asociados a las actividades para la entrada y salida de datos. La principal idea tomada de Case Handling es el uso de formularios de datos para la gestión del proceso, aunque nuestra propuesta se diferencia en que las actividades no se asocian a distintos formularios predefinidos, sino que los formularios son construidos dinámicamente para cada actividad en función de lo definido en OntoDD y OntoWF con la consiguiente ventaja en cuanto a reutilización e independencia de las distintas representaciones.

En las siguientes subsecciones se detallan los distintos elementos de nuestro modelo. En el resto del artículo, continuamos con el ejemplo de la solicitud de un préstamo por parte de un cliente de una entidad financiera para explicar la construcción y funcionamiento de alguno de los componentes.

3.1 OntoMetaWorkflow: ontología genérica para representación de workflows

OntoMetaWorkflow[4] contiene los elementos que componen los workflows para procesos de gestión administrativa y sus relaciones (ver Figura 2). Esta ontología se ha construido adaptando a las características de los procesos administrativos las definiciones de elementos proporcionadas por WfMC. En su realización se ha

[4] http://www.unex.es/eweb/project_weapon/OntoMetaWorkflow.owl

utilizado la metodología METHONTOLOGY [24] y se ha representado utilizando el lenguaje OWL.

Las clases que componen esta ontología son *AdministrativeProcess*, *RelevantData*, *Activity*, *WorkflowParticipant* y dentro de esta última la subclase *Root*. A continuación se explica cada clase junto con las propiedades de objetos y de datos que tienen definidas[5]:

- *AdministrativeProcess* (Proceso Administrativo) se utiliza para representar el proceso que se quiere gestionar con el workflow. Tiene definida la relación *consistsOf* con *Activity* para indicar las actividades que componen un proceso. También con *Activity* mantiene la relación *currentSituation* para almacenar en qué actividad se encuentra una instancia de un proceso. La relación *characterizedBy* con *RelevantData* se utiliza para referenciar los datos del dominio con los que trabajará el proceso. Por último, mediante la relación *generatedBy* con *WorkflowParticipant* se mantiene qué instancia de participante (usuario) inició la instancia del proceso. Además esta clase también tiene definidas las propiedades *beginningDate* para guardar la fecha de inicio de una instancia y *maxEndingDate* para guardar la fecha límite de finalización de una instancia del proceso.

- *RelevantData* (Datos Relevantes) es la clase que almacena datos comunes a todas las instancias de un proceso administrativo. Tiene definidas las propiedades *externalDocument* que estará a verdadero en el caso de que los datos hagan referencia a un documento externo, con la URL de dicho documento contenida en la propiedad *location*.

- *Activity* (Actividad) es la unidad lógica de trabajo. La relación *before* sirve para indicar la/s actividad/es que preceden inmediatamente a una actividad y la relación *after* para indicar la/s que va/n inmediatamente a continuación, conformando el flujo del proceso. Con la relación *isPerformedBy* se indica qué *WorkflowParticipant* puede realizar la actividad. Las propiedades *beforeControlFlowPattern* y *afterControlFlowPattern* permiten indicar, cuando existan enumeradas varias actividades en las relaciones *before* o *after* de una actividad, si son en paralelo (and) o son alternativas (xor). Las propiedades *showDomainData* y *fillInDomainData* permiten enumerar qué datos del dominio mostrará o habrá que seleccionar en una actividad respectivamente. De forma similar las propiedades *showProcessProperties* o *fillInProcessProperties* sirven para enumerar qué propiedades de tipo dato en OWL que tenga definidas un proceso en particular se van a mostrar o habrá que rellenar en dicha actividad. Por último, la propiedad *daysTimeFrame* sirve para indicar el plazo de días que hay para realizar una actividad, *dayNotice* los días de antelación para avisar al responsable y la propiedad *activityDescription* sirve para contener una descripción del trabajo a realizar en dicha actividad.

- *Workflow Participant* sirve para definir los usuarios que participan en el proceso. La relación *performs* sirve para indicar qué actividades puede hacer. Las propiedades *id* y *password* sirven para identificar en el sistema de workflow al participante y la propiedad *generator* se utiliza para indicar si el participante puede crear o no nuevas instancias del proceso. La subclase *Root* es una clase especial con privilegios de

[5] Siguiendo la nomenclatura de OWL, a las propiedades entre objetos las denominamos relaciones y a las propiedades de tipo dato las denominamos propiedades.

supervisor que puede crear nuevas instancias de proceso y realizar tareas de administración en el sistema.

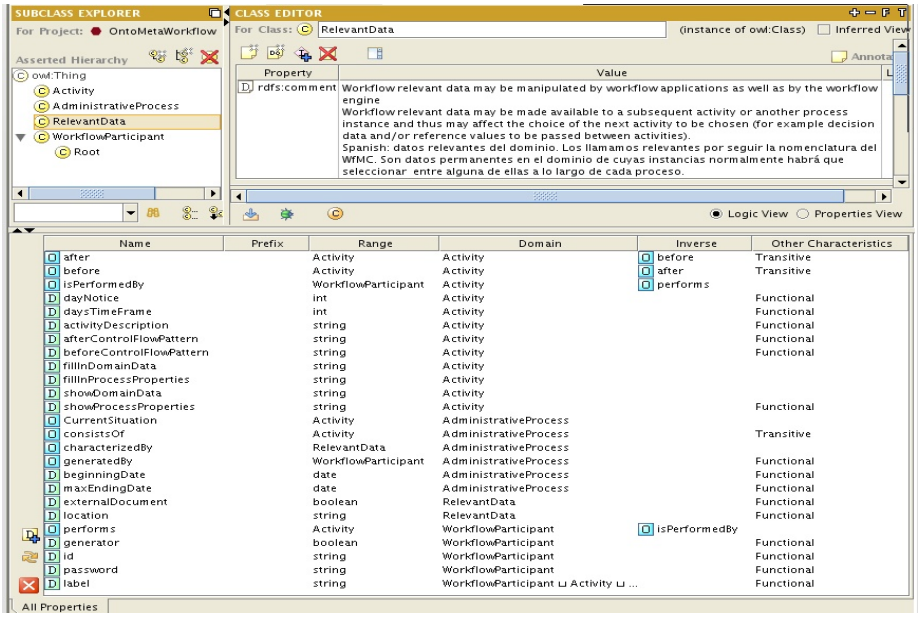


Fig. 2. Visualización en Protégé de OntoMetaWorkflow

3.2 OntoDD: ontología de datos relevantes al dominio

Esta ontología importa los elementos definidos en OntoMetaWorkflow y debe contener, por un lado, las taxonomías de los datos que serán usados en el dominio y, además, la taxonomía de los potenciales participantes en el workflow. La principal regla que debe cumplir OntoDD es que los elementos raíz de cada taxonomía estén enlazados con las superclases definidas en OntoMetaWorkflow, *RelevantData* y *WorkflowParticipant* (o *Root* en el caso de ser administrador del WfMS) respectivamente.

El desarrollo de esta ontología puede ser llevado a cabo de manera sencilla con cualquier aplicación que permita modelar ontologías en OWL, como Protégé, siguiendo estos pasos:

1. Importar OntoMetaWorkflow de manera que sus elementos serán superclase de los de OntoDD.
2. Identificar aquellos elementos del dominio comunes a todos los procesos para definirlos como subclases de *RelevantData* y definir sus instancias. Esto último es muy importante pues el WfMS trabajará con las instancias de estas clases. En el ejemplo, tendremos la clase de los tipos de préstamo con atributos como "cantidad máxima" o "plazo máximo de devolución" e instancias con los préstamos concretos que oferta esa entidad (personal, hipotecario, etc. junto con sus distintos plazos y cantidades).

3. Identificar las características de los distintos usuarios que utilizarán el workflow en clases del tipo *WorkflowParticipant* y crear (o reutilizar) las instancias por cada usuario concreto. Además, habría que indicar que alguna de esas clases es generadora y al menos una de las clases debería ser definida como subclase de *Root*. En el ejemplo bancario, podremos tener clases como "clientes del banco", "analistas de riesgo" o "responsables de la concesión del crédito" e instancias de usuarios de estos tipos. Siguiendo con este ejemplo, en la clase "cliente" tendremos el atributo *generator* a verdadero (ya que se crea una nueva instancia de proceso cuando un cliente solicita un préstamo) y la clase "responsable de crédito" como subclase de *Root*.

Es importante señalar que pueden ser reutilizadas tanto ontologías existentes bien formadas y completas de muchos dominios, o las ontologías que de sus usuarios puedan tener empresas o instituciones. Para que el sistema automáticamente trate con los elementos definidos en ella, sólo sería necesario poner dicha ontología como subclase de *RelevantData*, si la taxonomía es de datos del dominio o como subclase de *WorkflowParticipant* si la taxonomía es de usuarios del sistema. Como ejemplo, está disponible la ontología OntoDD para la gestión de solicitudes de préstamos[6].

3.3 OntoWF: ontología del workflow del dominio

Esta ontología representa el workflow del proceso administrativo que hay que gestionar. En ella se define el workflow junto con sus propiedades, las actividades de que consta, el orden de ejecución de dichas actividades, qué datos relevantes recogidos en OntoDD mostrará o modificará cada actividad y qué participantes pueden realizar cada actividad. Por tanto, para realizar esta representación se utilizan las especificaciones de elementos y relaciones posibles que puede tener un workflow y que están representadas en OntoMetaWorkflow, y los datos relevantes del dominio y los participantes posibles en el workflow definidos en OntoDD.

Para el diseño de la OntoWF de un proceso administrativo concreto se puede utilizar la herramienta WEAPON Designer que se detalla en la subsección 4.4. No obstante, aunque con mayor esfuerzo, es posible construir OntoWF manualmente sin necesidad de esta herramienta siguiendo los pasos que se detallan a continuación:

1. Importar la OntoDD del dominio y OntoMetaWorkflow.
2. Definir el proceso a gestionar como subclase de *AdministrativeProcess* y sus propiedades del tipo dato de OWL. Por ejemplo, en el dominio bancario tendremos una clase para el proceso de solicitud de préstamo junto con propiedades como los datos económicos del solicitante o la respuesta a dicha petición.
3. Definir cada actividad del proceso como subclase de *Activity* de OntoMetaWorkflow dando valores a todas las propiedades de la clase *Activity* que se exponen en la subsección 4.1

A modo de ejemplo está disponible la ontología OntoWF para la gestión de solicitudes de préstamos[7].

[6] http://www.unex.es/eweb/project_weapon/OntoDD_LoanApplication.owl
[7] http://www.unex.es/eweb/project_weapon/OntoWF_LoanApplication.owl

3.4 WEAPON Designer

Esta herramienta facilita la construcción de OntoWF utilizando la representación gráfica de WF-NET [25]). Esta herramienta va a permitir representar de forma gráfica las actividades del proceso, su orden de ejecución y, además, va a permitir relacionarlas con los datos y usuarios definidos en OntoDD en función de las propiedades definidas en OntoMetaWorkflow. Se ha implementado utilizando Jena[8] y WoPeD[9] como lienzo gráfico de definición de worflows.

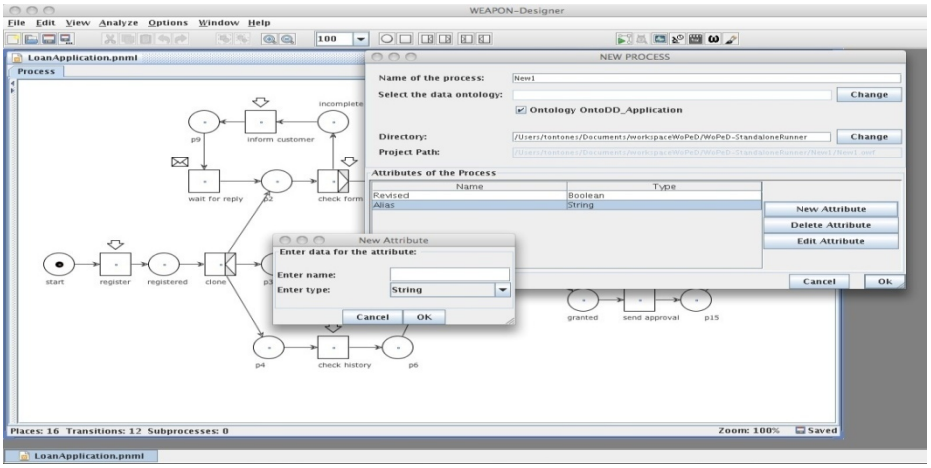


Fig. 3. Captura de WEAPON Designer

3.5 WEAPON Manager

WEAPON Manager es el motor que gestiona los procesos administrativos representados en las ontologías OntoWF y OntoDD a través de formularios web. La herramienta permite crear procesos administrativos concretos siguiendo las especificaciones de estas dos ontologías y crearlos como instancias de la clase *AdministrativeProcess* de OntoWF almacenadas en una base de datos. Es decir, en el caso del préstamo bancario, cada vez que se realiza una nueva solicitud de préstamo es almacenada en el sistema como instancia de la subclase de *AdministrativeProcess* que esté definida en la OntoWF de dicho proceso. Al igual que WEAPON Designer, se ha implementado utilizando el parser Jena y generando los formularios web en Java.

A diferencia de otros modelos de WfMS, WEAPON Manager puede reconstruir automáticamente la aplicación web que actúa como WfMS. Es decir, cualquier cambio que pueda producirse en el workflow del proceso administrativo que se está gestionando que afecte a los datos relevantes del proceso, a los usuarios involucrados o al número y orden de las actividades, puede ser especificado sin tener que modificar

[8] http://jena.sourceforge.net/
[9] http://www.woped.org

las otras partes. De esta forma, es posible modificar OntoWF, donde está definido el proceso, o modificar OntoDD, que contiene la información necesaria para las actividades, y WEAPON Manager regenera la aplicación web que actúa como WfMS. En la figura 4 se muestra la ventana de configuración de esta herramienta y en la figura 5 se muestra gestionando el workflow de solicitudes de préstamo.

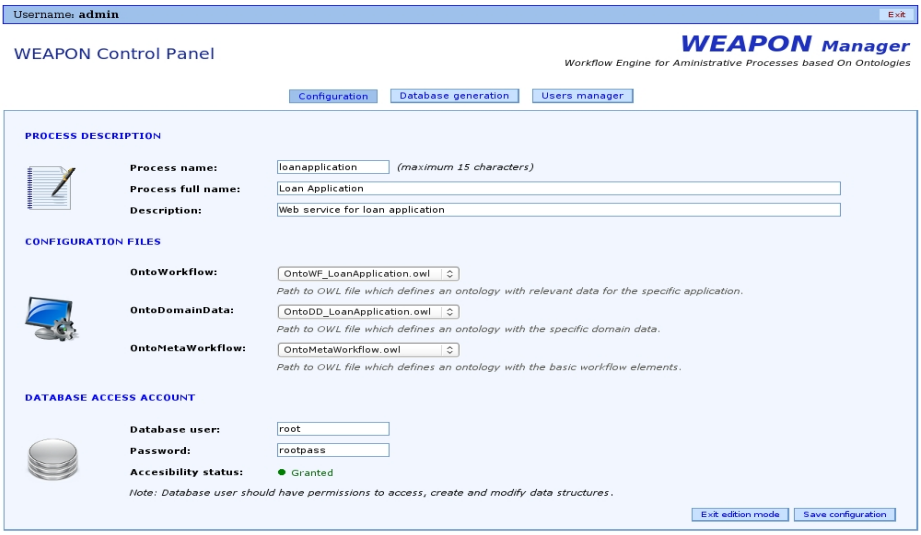


Fig. 4. Ventana de configuración de WEAPON Manager

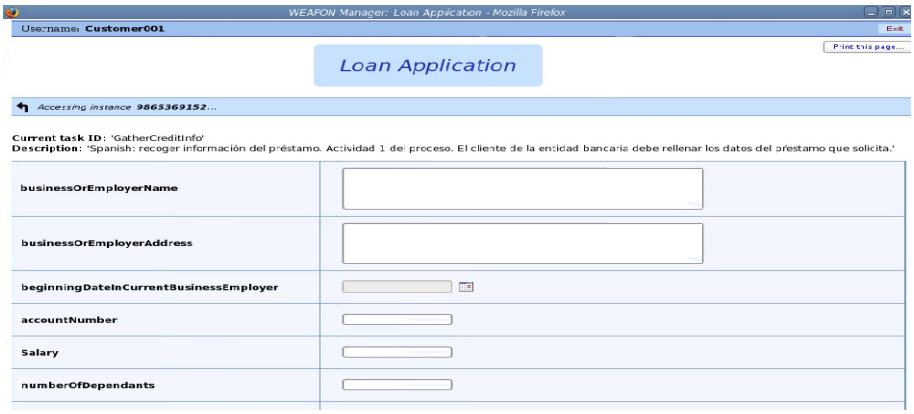


Fig. 5. Ejemplo de WEAPON Manager para un proceso de solicitud de préstamo

4 Conclusiones

Este trabajo utiliza las ontologías como soporte de representación para WfMS en el ámbito de la gestión administrativa. Debido a que el peso en este tipo de procesos

recae en la clasificación de los datos gestionados por las actividades y la categorización de los usuarios participantes en cada actividad, las ontologías proporcionan ventajas muy significativas como facilidad de uso, completitud e información consistente y consensuada en los datos y procesos. Además, su uso favorece la reutilización, adaptación e integración de los procesos y de los datos utilizados en cada actividad.

En este trabajo se ha presentado el modelo WEAPON (Workflow Engine for Administrative Processes based on ONtologies) que define cómo diseñar y explotar un proceso de negocio administrativo, apoyado por herramientas software. Su base es la ontología OntoMetaWorkflow que especifica los elementos y reglas que definen los workflows de acuerdo a los estándares y recomendaciones del WfMC, con las adaptaciones particulares que requiere el dominio de los procesos de gestión administrativa. Gracias a esta ontología se van a poder representar los datos relevantes al dominio y los usuarios involucrados en una ontología llamada OntoDD y las actividades que componen los procesos administrativos junto con los datos y usuario que necesitan para llevarse a cabo en una ontología llamada OntoWF.

Además, para facilitar la definición y gestión de procesos administrativos a partir de OntoMetaWorkflow, OntoDD y OntoWF se han implementado dos herramientas. Por un lado, WEAPON Designer que permite definir los workflows en la ontología OntoWF a partir de los elementos y restricciones descritos en OntoMetaWorkflow y OntoDD. Por otro, WEAPON Manager que permite generar la aplicación web que actuará como WfMS del workflow representado en OntoWF y OntoDD.

En la actualidad, este modelo está siendo utilizado por la empresa de servicios informáticos MPG Extremadura[10] en la gestión de servicios de incidencias TICs. Las opiniones de la empresa y usuarios que han utilizado WEAPON Designer indican que inicialmente cuesta identificar los procesos, usuarios y datos que intervienen pero, una vez el workflow ha sido definido y es gestionado en WEAPON Manager, es fácil reutilizarlo y adaptarlo para dominios similares.

Agradecimientos. Este trabajo se ha desarrollado dentro del proyecto TIN2008-02985 de CICYT y del proyecto PDT08A023 de la Junta de Extremadura.

Referencias

1. Gartner Group: Delivering IT's Contribution: The 2005 CIO Agenda. Gartner Inc (2005)
2. Workflow Management Coalition: Workflow Management Coalition Terminology & Glossary. Document No. WFMC-TC-1011, 3rd ed, WfMC, Winchester (1999)
3. McCready, S.: There is more than one kind of Workflow Software, Computerworld, November 2, pp. 85--90 (1992)
4. Georgakopoulos, D., Hornick, M.F., Sheth, A.P.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases 3, 2, pp. 119--153 (1995)
5. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, New York (2007)

[10] http://www.mpgex.es

6. Calero, C., Ruiz, F., Piattini, M: Ontologies for Software Enginnering and Software Technology. Springer-Verlag, Berlin (2006)
7. Vieira, T.A.S.C., Casanova, M.A., Ferrao, L.G.: An Ontology-Driven Architecture for Flexible Workflow execution. In: Joint Conference 10th Brazilian Symposium on Multimedia and the Web & 2nd Latin American Web Congress, pp. 70-77. IEEE Computer Society, Ribeirao Preto-SP, Brazil (2004)
8. Gasevic, D., Devedzic, V.: Petri net ontology. Knowledge-Based Systems, 19, 4, pp. 220--234 (2006)
9. Haller, A., Oren, E., Kotinurmi, P.: m3po: An ontology to Relate Choreography Models to Workflow Models. In: International Conference on Services Computing (SCC 2006), pp. 19--27. IEEE Computer Society, Chicago (2006)
10. Abramowicz, W., Filipowska, A., Kaczmarek, M., Kaczmarek, T.: Semantically enhanced Business Process Modelling Notation. In: Workshop on Semantic Business Process and Product Lifecycle Management, pp. 88--91. CEUR-WS, Innsbruck, Austria (2007)
11. Object Management Group: Business Process Modeling Notation (BPMN), Version 1.0. OMG Final Adopted Specification, OMG, Needham (2006)
12. Pathak, J., Caragea, D., Honavar, V.: Ontology-Extended Component-Based Workflows: A Framework for Constructing Complex Workflows from Semantically Heterogeneous Software Components. In: Bussler, C., Tannen, V., Fundulaki, I. (eds) SWDB Semantic Web and Databases, 2nd Int. Workshop, pp. 41--56. Springer, Toronto, Canada (2004)
13. Vidal, J.C., Lama, M., Bugarín, A.: A Workflow Modeling Framework Enhanced with Problem-Solving Knowledge. In: 10th Int. Conference on Knowledge-based intelligent information and engineering systems, pp. 623--632. Springer, Bournemouth (2006)
14. Yao, Z., Liu, S., Han, L., Reddy, Y.V.R., Yu, J., Liu, Y., Zhang, C., Zheng, Z.: An Ontology Based Workflow Centric Collaboration System. In: 10th International Conference Computer Supported Cooperative Work in Design III, pp. 689--698. Springer, Nanjing, China (2007)
15. Andonoff, E., Bouaziz, W., Hanachi, C.: A Protocol Ontology for Inter-Organizational Workflow Coordination. In: 11th East-European Conference on Advances in Databases and Information Systems (ADBIS), pp. 28--40. Springer, Varna, Bulgaria (2007)
16. Huang, N., Diao, S.: Ontology-based enterprise knowledge integration. Robotics and Computer-Integrated Manufacturing 24, 4, pp. 562--571 (2008)
17. BEA, IBM, Microsoft, SAP, Siebel: Business Process Execution Language for Web Services Version 1.1 (2003)
18. Workflow Management Coalition: Workflow Standard, Process Definition Interface -- XML Process Definition Language, Technical Report Document Number WFMC-TC-1025. Workflow Management Coalition, Winchester (2005)
19. van der Aalst, W., ter Hofstede, A.: YAWL: Yet Another Workflow Language. Information Systems, 30, 4, pp. 245--275 (2005)
20. Business Process Management Initiative: Business Process Modeling Language. Specification Version 1.0. BPMI.org, Needham (2002)
21. Lozano-Tello, A., Gómez-Pérez, A.: Applying the ONTOMETRIC Method to Measure the Suitability of Ontologies. In: Green, P., Rosemann, M., (eds.) Business Systems Analysis with Ontologies, Chapter IX, pp.249—269. Idea Group Publising, Hershey (2005)
22. Dang, J., Hedayati A., Hampel K., Toklu, C.: An ontological knowledge framework for adaptive medical workflow. Journal of Biomedical Informatics 41, 5, pp. 829--836 (2008)
23. van der Aalst, W., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process. Data & Knowledge Engineering 53, 2, pp. 129--162 (2005)
24. Gomez-Perez, A., Corcho, O., Fernandez-Lopez, M.: Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer (2004)
25. van der Aalst, W., van Hee, K.: Workflow Management - Models, Methods and Systems. Advanced Information and Knowledge Processing/MIT Press, Cambridge (2002)

Populating Data Warehouses with Semantic Data

Victoria Nebot and Rafael Berlanga

Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
{romerom, berlanga}@uji.es

Abstract. The Semantic Web has become a new environment that enables organizations to attach semantic annotations taken from domain and application ontologies to the information they generate. As a result, large amounts of complex, semi-structured and heterogeneous semantic data repositories are being made available. In this paper, we present an automatic method for on-demand construction of OLAP cubes aimed at analyzing semantically enriched data expressed as instance stores in RDF/OWL. The starting point of the method is a simple multidimensional star schema (i.e. topic of analysis, dimensions and measures) designed by the analyst by picking up concepts and properties from the ontology of the instance store. The method uses an interval indexing scheme over such an ontology in order to handle basic entailments. Moreover, the intrinsic graph structure of the instances is treated using composition paths among the analysis topic and dimension instances. These paths allow us to combine instance values in an appropriate way to construct the fact table. Later on, actual data analysis can be performed using standard OLAP tools. We have tested our approach with a synthetically generated instance data in order to show its scalability and effectiveness.

Key words: Semantic Web, Data Warehouses, ETL processes, Instance Stores.

1 Introduction

Data analysis tools, such as OLAP (On-Line Analytical Processing) [3] tools allow decision makers to efficiently extract useful information from large amounts of transactional data. These tools rely on the multidimensional (MD) structures of a data warehouse (DW) such as facts and dimension hierarchies that allow the analyst to explore and aggregate information at different levels of detail. Although OLAP analysis has gained widespread acceptance in traditional business applications, data with richer structure other than relational data is being generated and made available, which brings the challenge of adapting and using OLAP tools over this new type of sources [12, 4].

The Semantic Web has become mature enough to provide different technologies and tools to generate semantic data. Nowadays many applications (e.g., medical applications) attach metadata and semantic annotations to the information they produce, (e.g. medical images, laboratory tests, etc). Semantic annotations are formal descriptions of information resources which usually rely on widely accepted domain ontologies. The main reason for using domain ontologies is to set up a common terminology and semantics for the concepts involved in a particular domain. Semantic annotations

are especially useful for describing unstructured, semi-structured and text data, which cannot be managed properly by current database systems. Currently, large repositories of semantically annotated data are being available (e.g. DBpedia [5], BioRDF [2], ...) opening new opportunities for enhancing current decision support systems.

The main purpose of this paper is to propose an automatic method for on-demand transforming and combining large amounts of semantically enriched instance data into a MD structure suitable for OLAP analysis. Such an approach includes both the design and the population of the MD schema. Our previous work [9] has tackled the design phase, for which we propose a formal-based method for an analyst to define the MD model by identifying the topic of analysis, the measures and dimensions of a data set. Other approaches such as [13] try to automate the MD design of DWs from a domain ontology. Either way, the method we present in this paper focuses on automating the population of the fact table from the instance data when the MD model has already been proposed. We leave the population of the dimension hierarchies for future work. In order to accomplish this task in an efficient way we introduce some index structures over the ontology schema and the instance data. As a result, our method can be considered an ETL (Extract, Transform and Load) process for populating a data warehouse with semantic data.

This approach raises new challenges with regard to current automated population of DWs. We cannot assume anymore that data sources are implemented over relational DBs and we need to focus on the application ontologies representing our data sources. Ontologies are semantically richer than relational schemas and the population process will be guided by knowledge expressed in the ontology, avoiding to perform exhaustive pattern searches over data. The approach in [10] uses semantic web technologies to populate OLAP cubes. They use ontology maps to convert the data sources to RDF [1] and then query this RDF data with SparQL [2] to populate the OLAP schema. The inconvenient of this approach is that users have to be schema-aware to query RDF data with SparQL, and it can become a cumbersome task when the data has a rich and heterogeneous structure. In contrast, our method automatically extracts and combines instances accordingly to create valid fact rows.

The rest of the paper is organized as follows. Section 2 describes an application scenario that motivates our approach. Section 3 contains the foundations of the approach. In Section 4 we give details about the method proposed. Section 5 shows the performance evaluation and Section 6 gives some conclusions and future work.

2 Application Scenario

The application scenario selected is Biomedicine in which vast amounts of semantically annotated data are being generated by many different types of data management systems [6]. In order to guide the process of semantically annotating the data, current data management systems adopt specific application ontologies relying on one or more widely accepted domain ontologies. A domain ontology in the biomedical scenario is

[1] RDF, Concepts and Abstract Syntax: `http://www.w3.org/TR/rdf-concepts/`, 2004.

[2] SparQL for RDF: `http://www.w3.org/TR/rdf-sparql-query/`, 2008

generally composed by a large corpus of semantically related data that describes the vocabularies and knowledge agreed by the relevant biomedical community.

In this scenario, semantically annotated data consists of many different types of data (e.g. lab test reports, ultrasound scans, images, etc.) steming from heterogeneous data sources. This data also presents complex relationships that evolve rapidly as new biomedical research methods are applied. As a consequence, this data cannot be properly managed by current data warehouse technology, mainly because it is complex, semi-structured, dynamic and highly heterogeneous.

Figure 1 illustrates an ontology fragment for the Rheumatology domain. As the figure shows, a patient may have different reports, consisting of the results of some blood tests (defined in the GALEN[3] ontology) and rheumatologic exams, the diagnosis of a disease (defined in the domain NCI[4] ontology) and the proposed treatment. The treatment is modeled as a collection of drug therapies (defined according to the UMLS [5] ontology). The patient has also a genetic profile. The genes involved in the genetic profiles are described by using the GO[6] domain ontology.

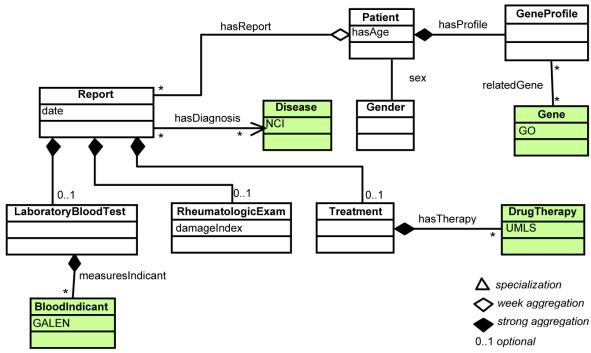


Fig. 1. A fragment of an application ontology for Rheumatology

Although in Figure 1 we have used UML to graphically represent the ontology fragment, the actual representation formalism is RDF/(S)[7] or OWL[8]. External concepts coming from domain ontologies are represented in the UML diagram with shaded boxes, indicating the source ontology within the attribute section (e.g. NCI, GO, etc.) Domain ontologies can be used to control the vocabulary and to bring further semantics to the annotated data [8]. Figure 2 shows the underlying graph structure of some OWL instances generated from the application ontology of Figure 1. This toy example will be

[3] GALEN: `http://www.opengalen.org/`
[4] NCI: `http://www.nciterms.nci.nih.gov/`
[5] UMLS: `http://www.nlm.nih.gov/research/umls/`
[6] GO: `http://www.geneontology.org/`
[7] RDF Schema: `http://www.w3.org/TR/rdf-schema/`, 2004.
[8] OWL Web Ontology Language: `http://www.w3.org/TR/owl-features/`, 2004.

used and developed throughout the paper. As it can be observed, instances (i.e. nodes) are related to each other through properties (i.e. arrows), resulting in a complex and heterogeneous graph-like structure. In particular, the figure shows one instance of *Patient* along with the corresponding features and relations to other instances. This example, although short and simple, reflects very well the complexity and heterogeneity that instance data can reach, in comparison to source data in relational databases of traditional warehouses.

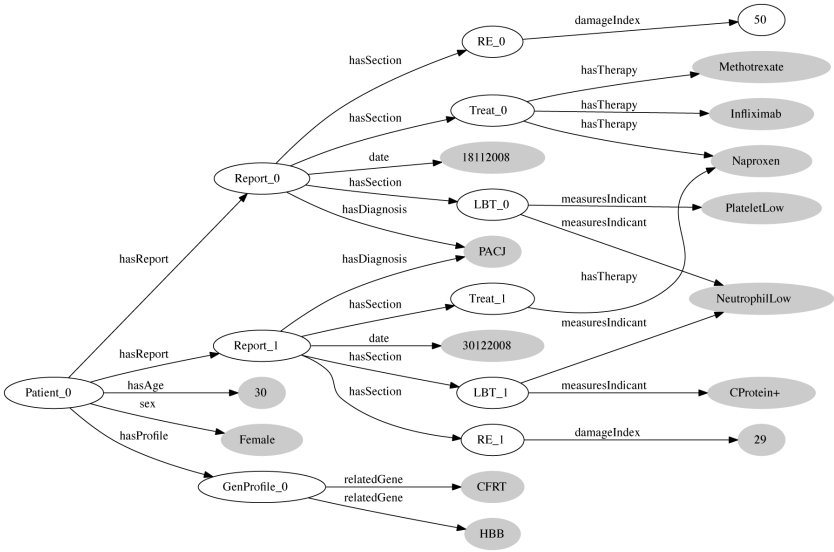


Fig. 2. Example of OWL instances generated from AO of Figure 1

Given a large repository of semantic instances of the kind shown in Figure 2, our aim is to build an automatic method to extract and combine only the instances relevant to the analyst multidimensional model. The target instances are processed and combined using their implicit semantics and structure to populate the fact table of a DW, which can be later analysed using OLAP-based techniques. As earlier mentioned, this paper is not concerned about the multidimensional model design. Therefore, we assume the design process has already been carried out either by the analyst or automatically. Following with the example, the subject of analysis is *Patient* and the different dimensions are: the patient's *age* and *gender*, the *diagnosis*, the *biomarkers* taken from the patient, the *damage index* of the patient's joints and the *drugs* administered during the *follow-up visits*.

A previous step to the ETL process consists of mapping the multidimensional schema of the analyst to the application ontology. Such mapping is not needed in case the multidimensional schema is automatically built from the ontology, since the different dimensions and measures already correspond to concepts and properties in the ontology.

Otherwise, the mapping has to be performed. In order to simplify this process, we allow the user to only select the potential measures and dimensions from a manually preselected subset of concepts and properties of the ontology. For the running example, the mappings among the general multidimensional schema and the application ontology concepts and properties of Figure 1 are shown in Table 1. From now on, we will use the term *dimension* to refer to the concepts and properties in the application ontology the analyst dimensions map to.

MD Schema	Ontology concepts and properties
Age	hasAge (data type property)
Gender	Gender (concept)
Diagnosis	Disease (concept)
Biomarkers	BloodIndicant, Gene (concepts)
Damage Index	damageIndex (data type property)
Follow-up visit	date (data type property)
Drug	DrugTherapy (concept)

Table 1. Mapping between Multidimensional Schema and Application Ontology

3 Foundations

In this section we present the concepts and definitions that give support to the method we have developed.

Definition 1 (Multidimensional Ontology Schema). *We define a Multidimensional Ontology Schema as a two-tuple* $S = (F, D)$ *where* F *is the fact type and* $D = \{T_i\}_{i=1,\ldots,m}$ *is its corresponding dimension types. The dimension types* T_i *are concepts and properties resulting from the mapping of the analyst initial dimension types to the ontology entities.*

Table 1 shows the dimensions in the application ontology of Figure 1 for the multidimensional schema selected. The fact type is *Patient* and $D = \{$ *hasAge*, *Gender*, *Disease*, *BloodIndicant*, *Gene*, *damageIndex*, *date*, *Drug*$\}$.

Definition 2 (Focus instance). *The focus instance is an instance belonging to the subject of the analysis or fact type. The focus instance must have a correspondence to a concept of the application ontology (focus concept).*

In the running example, we will have *Patient* as the fact type and also focus instance, since it maps to the concept *Patient* in the application ontology of Figure 1.

The granularity of the fact table is the lowest level of information that will be stored in the fact table. It determines which dimensions will be included and where along the hierarchy of each dimension the information will be kept. We want to include all the dimensions specified by the analyst and we consider the information found in the ontology to be at the lowest level in the dimension hierarchies. Therefore, we define a fact as follows:

Definition 3 (Fact). *We define a fact as a* valid combination *of dimension values that co-occur within a focus instance.*

We will later define *valid combination* but for now, lets say that every possible combination of dimension values co-occurring in a focus instance is not admissible. For example [11], in a retail business where data sources are relational, the typical focus instance would be *purchase* and one instance of purchase can be described by one fact containing the dimension values of the *location* of purchase, the type of *product* and the *time* of the purchase. However, the rich graph-like structure of our focus instance implicitly restricts the valid combinations with other instances and literals. In order to automatically find out the valid combinations of entities within a focus instance we need to introduce some definitions. From now on, the definitions are applied to just one focus instance, which besides is assumed to be a DAG (directed acyclic graph).

Definition 4 (Context of two dimensions/ dimension values). *Given two dimensions T_i and T_j we define the context of T_i and T_j, $context(T_i, T_j)$, as the nearest common ancestor concept of T_i and T_j inferred from the ontology. Similarly, the context of two dimension values $v_i \in T_i$ and $v_j \in T_j$ is the nearest common ancestor instance of v_i and v_j in the ontology instance data.*

For example, the context of dimensions $Disease$ and $damageIndex$ is $Report$, and the context of dimension values $PACJ$ and 29 is $Report\_1$.

Definition 5 (Independent/dependent dimensions). *Given two dimensions T_1 and T_2, they are considered independent dimensions if their context is the focus instance. Otherwise, the dimensions are dependent.*

For example, $hasAge$ and $Gender$ are independent dimensions (their context is *Patient*), whereas $Disease$ and $damageIndex$ are dependent (their context is *Report*).

Definition 6 (Context dependent). *Given two dimension values v_i and v_j, $Type(v_i) = T_i,\ Type(v_j) = T_j,\ T_i \neq T_j,\ T_i, T_j \in D$ co-occurring within a focus instance, v_i and v_j are context dependent if $Type(context(v_i, v_j)) = context(T_i, T_j)$.*

The previous dimension values $PACJ$ and 29 are context dependent because their context is $Report\_1$, whose type is $Report$ and coincides with the context of their respective dimensions, $Disease$ and $damageIndex$.

Definition 7 (Valid combination). *We define a valid combination of dimension values of an instance as a tuple $(v_1, v_2, ..., v_n)$ where $n = |D|$ and $\forall(i, j), (v_i, v_j)$ $are\ context\ dependent$.*

As an example of the previous definition, dimension values *30122008* and *CProtein+* are a valid combination because they are context dependent. However, values *30122008* and *Infliximab* are not a valid combination because they are not context dependent, that is the context of their dimensions (i.e. *Report*) does not coincide with the type of their context (i.e. *Patient*).

4 Overview of the approach

We start with an overview of the different steps involved in our method, which is depicted in Figure 3. As it can be seen, there is an *off-line phase*, which creates some indexes over the target ontology to speed-up the *on-line phase*, which automatically populates a fact table from a multidimensional model with the ontological instances of the knowledge repository.

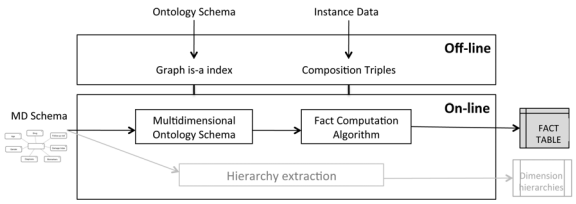


Fig. 3. Architecture of the method proposed

On-line phase. In order to compute valid facts, the multidimensional schema of the analyst is mapped to concepts and properties of the application ontology. As a result, we obtain a Multidimensional Ontology Schema as specified in Definition 1. Then, we apply the fact computation algorithm, which is in charge of grouping dimensions and combining dimension values within a group and across groups. Finally, from the combinations of dimension values obtained we filter only the valid combinations according to Definition 7.

Off-line phase. In order to perform the previous steps in an efficient way we preprocess both the ontology schema and instances. Regarding the instances, we create the *composition triples*, which make efficient the retrieval of all the instances and values reachable from an instance through the composition of properties. The composition triples are enriched with the *graph is-a index* so that we can retrieve both asserted and inferred instances of a concept. The *graph is-a index* results from applying an interval labeling scheme to the extended "is-a" graph of the ontology, in which implicit "is-a" relationships other than the asserted ones have been made explicit. This way each node has encoded ancestors and descendants as intervals. More details about the interval labeling scheme are given in [7]. For example, if we want to retrieve all *Patient* instances along with the diseases they have, we will be able to retrieve instances whose type is *Patient* as well as all the instances that are classified under it, such as *RheumaticPatients*, *CardioPatients*, etc. along with the corresponding diseases classified under *Disease*. Therefore, basic instance reasoning is performed without needing a reasoner.

4.1 Composition triples

Following with the running example, for the focus instances (e.g. instances of *Patient*) we want to find the different dimension values that characterize them. In order to do that, we define a specific type of triples, which we call *composition triples*.

Definition 8 (Composition triples). *A composition triple is a statement of the form (subject, predicate, object) where the subject is a resource, the object is either a resource or a data type and the predicate is a composition path from the subject to the object. A composition path is an alternate sequence of properties and resources that connect both the subject and the object.*

We can see the composition triples enriched with the "is-a" index of *Patient_0* to the dimension values in Table 2. The fourth column refers to the "is-a" index and what is actually recorded is a foreign key to the concept the object instance belongs to. Similarly, there is another column for the subject instance. Currently, this table is generated when parsing the RDF/OWL file describing the instances by traversing the graph of each instance (e.g. graph of Figure 2).

Subject	Composition Path	Object	Dimension
Patient_0	/sex	Female	Gender
Patient_0	/hasAge	30	hasAge
Patient_0	/hasProfile/GeneProfile_0/relatedGene	HBB	Gene
Patient_0	/hasProfile/GeneProfile_0/relatedGene	CFRT	Gene
Patient_0	/hasReport/Report_0/date	18112008	date
Patient_0	/hasReport/Report_0/hasDiagnosis	PACJ	Disease
Patient_0	/hasReport/Report_0/hasSection/LBT_0/measures...	PlateletLow	BloodIndicant
Patient_0	/hasReport/Report_0/hasSection/LBT_0/measures...	NeutrophilLow	BloodIndicant
Patient_0	/hasReport/Report_0/hasSection/RE_0/damageIndex	50	damageIndex
Patient_0	/hasReport/Report_0/hasSection/Treat_0/hasTherapy	Naproxen	DrugTherapy
Patient_0	/hasReport/Report_0/hasSection/Treat_0/hasTherapy	Methotrexate	DrugTherapy
Patient_0	/hasReport/Report_0/hasSection/Treat_0/hasTherapy	Infliximab	DrugTherapy
Patient_0	/hasReport/Report_1/date	30122008	date
Patient_0	/hasReport/Report_1/hasDiagnosis	PACJ	Disease
...	...	...	...

Table 2. Composition triples of the focus instance to each of their dimension values.

4.2 Fact Computation Algorithm

The algorithm to obtain valid combinations of dimension values performs two main steps: partition of dimensions into independent groups, and combination of dimension values within and across groups. Let D be the set of dimensions. G is a partition of D formed as follows:

- $\forall g \in G,\ \forall\ T_i, T_j \in g,\ \ T_i\ is\ dependent\ of\ T_j$
- $\nexists\ g, g\prime \in G\ such\ that\ T_i \in g, T_j \in g\prime\ and\ T_i\ is\ dependent\ of\ T_j$
- $\forall g \in G,\ context(g) = context(T_1, ..., T_n)\ such\ that\ T_i \in g, n = |g|$

Each group g in the previous partition G holds dependent dimensions and the context of each group is the context of the set of dimensions in the group. This partition can be efficiently calculated by checking dependencies among dimensions using the composition paths of the instances.

For the running example, the partition G of dimensions along with the context of each group (following the syntax $Group_{context} < dimensions >$) is as follows:

- $G1_{Patient} < Gender >$
- $G2_{Patient} < hasAge >$
- $G3_{GeneProfilet} < Gene >$
- $G4_{Report} < date, damageIndex, Disease, BloodIndicant, DrugTherapy >$

The second step of the algorithm consists of the combination of dimension values within and across the previous groups of dimensions for each focus instance. This is performed by applying the following relational algebra operation over the composition triples:

$$\bowtie_{g_i \in G} (\sigma_{valid\_comb.}(\bigcup_{I \in Inst(context(g_i))} \times_{T_j \in g_i} \Pi_{(T_j)} I))$$

Thus, for each instance I of the context of a group, we perform the cartesian product of the tuples resulting from projecting each dimension of the group over the instance I. The resulting tuples are filtered through the selection operator that selects just the tuples that are valid combinations according to Definition 7. The previous operations are performed over each group of the partition G and finally, we join the obtained tuples for each of the groups. Notice $\Pi_{(T_j)} I$ is the projection of the instance I over the dimension T_j, which is efficiently performed thanks to the composition triples.

Patient	**G1**	**G2**	**G3**	**G4**				
	Gender	hasAge	Gene	date	damageIndex	Disease	BloodIndicant	DrugTherapy
Patient_0	Female	30	HBB	30122008	29	PACJ	Cprotein+ NeutrophilLow	Naproxen
			CFRT	18122008	50	PACJ	NeutrophilLow PlateletLow	Naproxen Infliximab Etacernept

Table 3. Groups of dimensions and dimension values for focus instance *Patient_0*

The results of applying the previous formula to the running example are shown in Table 3. The table shows the dimension values without the cartesian product of dimension values inside a group and the join of groups. As we can see, the generation of dimension values for groups $G1$, $G2$ and $G3$ is trivial, since each group is composed by only one dimension. For group $G1$ we project dimension $Gender$ over instance *Patient_0*, obtaining $Female$. For group $G2$ we project $hasAge$ over instance *Patient_0*, obtaining 30. For group $G3$ we project $Gene$ over instance *GeneProfile_0*, obtaining HBB and $CFRT$. For group $G4$, its context is $Report$ and there happen to be two instances of $Report$, *Report_0* and *Report_1*, in *Patient_0* so, first we project each dimension of the group over *Report_1*, obtaining the dimension values shown in the first row of group $G4$, and then over *Report_0*, obtaining the values of the second row of $G4$.

After applying the cartesian product of the dimension values for each row in each group we obtain combinations in each group that may not be valid. In order to filter valid combinations the operator of selection is used over the tuples of each group. Finally, the join of each group is performed using the focus instance as key. In the example, all combinations that will be generated from Table 3 are valid and the result of joining the tuples in each group are shown in Table 4.

FACT TABLE								
Focus inst.	**Gender**	**hasAge**	**Gene**	**date**	**dIndex**	**Disease**	**BloodIndicant**	**DrugTherapy**
Patient_0	Female	30	HBB	30122008	29	PACJ	CProtein+	Naproxen
Patient_0	Female	30	HBB	30122008	29	PACJ	NeutrophilLow	Naproxen
Patient_0	Female	30	HBB	18122008	50	PACJ	NeutrophilLow	Naproxen
Patient_0	Female	30	HBB	18122008	50	PACJ	NeutrophilLow	Infliximab
Patient_0	Female	30	HBB	18122008	50	PACJ	NeutrophilLow	Etacernept
Patient_0	Female	30	HBB	18122008	50	PACJ	PlateletLow	Naproxen
Patient_0	Female	30	HBB	18122008	50	PACJ	PlateletLow	Infliximab
Patient_0	Female	30	HBB	18122008	50	PACJ	PlateletLow	Etacernept
Patient_0	Female	30	CFRT	30122008	29	PACJ	CProtein+	Naproxen
...	...	...	...	...	...	...	...	

Table 4. Facts generated for Patient_0

5 Evaluation

In order to evaluate our proposal, we have designed an experiment with the patient application ontology of Figure 1, which is a simplification of an application ontology developed in the Health-e-Child[9] project. The schema of this ontology contains 245 classes and 15 properties. Instances are synthetically generated from the features of a small set of real patients. For this purpose, we have extended the XML generator presented in [14] with new operators specific for RDF/OWL formats, namely: a generator for identifiers and several RDF/OWL attribute fillers. Composition relationships between instances are then generated with the existing XML generator operators (e.g. `sequence`, `xor` and `subPattern`), but expressing the result as values of the `rdf:about` and `rdf:resource` attributes. As a result, we are able to generate synthetic instance data of any size and with the structural variations we want to model. In order to test the efficiency of our approach we have generated 3000 instances of patient with heterogeneous structure. Each patient has between one and three altered genes, between one and five reports, between one and three blood biomarkers and between one and three treatments. The total amount of instances generated results in 188.742 instances.

Figure 4 shows the performance evaluation of the method with the previous synthetic ontology. Given the set of dimensions of the running example, we have generated a cuboid for each of the possible subsets of the given dimensions. The result would form a lattice of cuboids, each showing the data at a different level of summarisation. In

[9] `http://www.health-e-child.org/`

particular, we have computed all the cuboids that can be generated with 8 dimensions and have organized them according to the number of dimensions involving each cuboid (x axis), from one dimension to eight. Axis y shows the time performance in seconds. As we can see, the time complexity is linear with respect to the number of dimensions of the cuboid, which proves the scalability and efficiency of the approach.

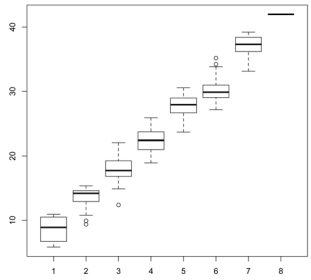


Fig. 4. Generation of cuboids performance w.r.t. the # of dims. involved in the cuboid.

6 Conclusion and Future Work

We have presented a preliminary approach to automatically extract and properly combine semantic instances expressed in RDF/(S) and OWL formats into a fact table of a DW. To our knowledge, this is the first method addressing this issue from ontological instances. Hence, we do believe this work opens new interesting perspectives as it combines the DW and OLAP techniques with the Semantic Web area. The method has proved to be scalable when dealing with complex and heterogeneous instance stores. In the future, new techniques for indexing the data will be considered for further speed up. Also, we would like to investigate new techniques for the correct aggregability of measures over the resulting fact tables, as well as the population of the dimension hierarchies from semantic sources. Another open research line that needs further investigation is the mapping of the analyst multidimensional schema to the application ontology when the former has been designed independently.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., and Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications In: Description Logic Handbook. Cambridge University Press 2003.
2. BioRDF Project: `http://esw.w3.org/topic/BioRDF_Top_Level_Task`

3. Chaudhuri, S., and Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. 1997, 26(1):65–74.
4. Chen, C., Yan, X., Zhu, F., Han, J., Yu, P.S.: Graph OLAP: Towards Online Analytical Processing on Graphs. ICDM 2008, 103–112
5. DBpedia Project. `http://dbpedia.org`
6. Garwood, K., McLaughlin, T., Garwood, C., Joens, S., Morrison, N., Taylor, C. F., Carroll, K., Evans, C., Whetton, A. D., Hart, S., Stead, D., Yin, Z., Brown, A. J., Hesketh, A., Chater, K., Hansson, L., Mewissen, M., Ghazal, P., Howard, J., Lilley, K. S., Gaskell, S. J., Brass, A., Hubbard, S. J., Oliver, S. G., and Paton, N. W.: PEDRo: a database for storing, searching and disseminating experimental proteomics data. BMC Genomics 2004, 5(68).
7. Nebot, V., and Berlanga, R: Efficient Retrieval of Ontology Fragments Using an Interval Labeling Scheme. In: 13th edition of the Spanish Conference on Software Engineering and Databases (JISBD) 2008.
8. Nebot, V., and Berlanga, R: Building tailored ontologies from very large knowledge resources. In: 11th Int. Conference on Enterprise Information Systems (ICEIS) 2009.
9. Nebot, V., and Berlanga, R., Pérez, J. M., and Aramburu, M.J.: Multidimensional Integrated Ontologies: A Framework for Designing Semantic Data Warehouses. In: Journal of Data Semantics, JoDS XIII: Special Issue "Semantic Data Warehouses", copyright (c) Springer, 2009.
10. Niinimaki M., and Niemi, T.: An ETL Process for OLAP Using RDF/OWL Ontologies. In: Journal of Data Semantics, JoDS XIII: Special Issue "Semantic Data Warehouses", copyright (c) Springer, 2009.
11. Pedersen, T. B., Jensen, C. S., and Dyreson, C. E.: A foundation for capturing and querying complex multidimensional data. Information Systems 2001, 26(5): 383–423.
12. Pérez, J.M., Berlanga, R., Aramburu, M.J., Pedersen, T.B.: Integrating Data Warehouses with Web Data: A Survey. IEEE Trans. Knowl. Data Eng. 2008,20(7):940-955
13. Romero, O., and Abello, A.: Automating multidimensional design from ontologies. In: Proc. of the 10th Int. Workshop on Data Warehousing and OLAP, pp. 1–8, 2007.
14. Sanz, I., Mesiti, M., Guerrini, G., and Berlanga, R.: Fragment-based approximate retrieval in highly heterogeneous XML collections. Data Knowl. Eng. 2008, 64(1): 266–293.

Towards the Semantic Desktop: the *seMouse* approach

Jon Iturrioz, Oscar Díaz, and Sergio F. Anzuola

ONEKIN Research Group
University of the Basque Country
San Sebastián, Spain
jon.iturrioz@ehu.es,oscar.diaz@ehu.es,jibfeans@si.ehu.es

Abstract. The semantic desktop aims at handling files as resources of a desktop ontology. This ontology captures a mental model of how the user conceptualizes files, and serves to re-interpret current file operations in a semantic manner. Now, a ".doc" artifact is not a mere file but a resource of the knowledge base. The user no longer creates a ".doc" file but an instance of, for instance, the Deliverable ontological class. Likewise, zipping operations do not require selecting one-at-a-time the files to be zipped together but semantic associations can be automatically traversed to locate the resources to be zipped. This paper presents seMouse, a realization of this vision where the mouse acts as a bridge between the traditional file world and the ontology realm. This accounts for a dual perspective of desktop resources. As files, traditional tooling is available (e.g. the Word editor). As resources, *the semantic mouse* permits to re-interpret traditional file operations from a semantic perspective whereby resource edition, classification, copying or zipping benefit from the underlying ontology.

Abstract of the paper included in the IEEE Intelligent Systems Journal, Vol. 23, Issue 1, pp. 24-31, 2008

Logic-based Ontology Integration using ContentMap

Ernesto Jiménez-Ruiz[1*], Bernardo Cuenca Grau[2],
Ian Horrocks[2], and Rafael Berlanga[1]

[1] Universitat Jaume I, Spain, {ejimenez,berlanga}@uji.es
[2] University of Oxford, UK, {berg,ian.horrocks}@comlab.ox.ac.uk

Abstract We present ContentMap, a system that uses a general method and novel algorithmic techniques to facilitate the integration of independently developed ontologies using mappings. Our method and techniques aim at helping users understand and evaluate the semantic consequences of the integration, as well as to detect and fix potential errors.

1 Motivation

Ontology integration techniques are often needed both during ontology development (e.g., when an ontology being developed reuses one or more external ontologies), and when ontologies are used in conjunction with data (e.g. when integrating and querying data sources annotated using different ontologies).

When the ontologies to be integrated have been independently developed, their vocabularies will most likely diverge, either because they use different names or naming conventions to refer to their entities. *Ontology Matching Techniques* [1] are intended to automatically discover the correspondences (i.e. *mappings*) between entities of different ontologies. This task is rather hard since in most cases there is not a common/similar nomenclature for the entity names.

Once the mappings (manually or automatically) have been generated the ontologies can be integrated. At this point, errors due to semantic incompatibility, may arise. There are two main causes for these errors. On the one hand, mappings suggested by automated tools are likely to include some errors. On the other hand, even if the correct mappings have been found (e.g. gold standard mappings), the ontologies may contain conflicting descriptions of the overlapping entities. These errors manifest themselves as unintended logical consequences (e.g. unsatisfiable concepts or wrong inferences), and they can be difficult to detect, understand and repair.

In [2] we presented a new general method and novel algorithmic techniques to obtain the right logical consequences when integrating ontologies using mappings. In this paper we focus on ContentMap (A logiC-based ONtology inTEgratioN Tool using MAPpings), the system we developed to provide just such support.

*This work has been partially supported by MICINN (TIN2008-01825/TIN) and Generalitat Valenciana (BPI06/372).

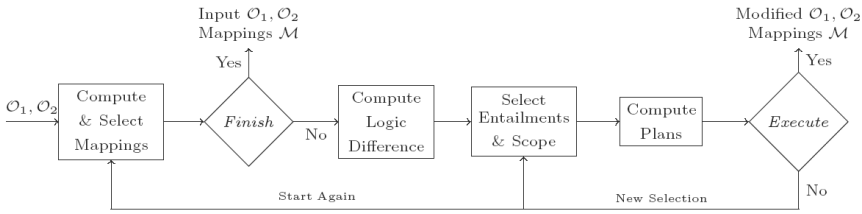


Figure 1. ContentMap Integration Method

2 Ontology Integration Method

We assume from now on that a set of mappings is represented as an OWL 2 [3] ontology $\mathcal{M}$, where mappings (often expressed as a tuple $\langle id, e_1, e_2, n, \rho \rangle$ [1]) are given as OWL 2 axioms of the form SubClassOf(e_1, e_2), EquivalentClasses(e_1, e_2), or DisjointClasses (e_1, e_2), for ρ of the form $(\sqsubseteq)$, $(\equiv)$, or $(\bot)$ respectively. e_1, e_2 are entity names in the vocabulary of $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively, id is a unique identifier for the mapping, and n is a numeric confidence measure between 0 and 1 which is represented as an annotation axiom [3] of the mapping axiom.

Figure 1 shows the ontology integration method followed in ContentMap to evaluate and repair the logic consequences of merging two independent ontologies using mappings. The method can be split as follows:

1. Compute mappings $\mathcal{M}$ between $\mathcal{O}_1$ and $\mathcal{O}_2$ using a mapping algorithm, and filter them according a given criteria.
2. Compute logic difference and evaluate impact by comparing the entailments holding before and after the integration.
3. Detect unintended entailments and select them.
4. Compute repair plans and execute best one according to the user necessities.

3 Underlying Techniques and ContentMap Support

Next we briefly comment the main underlying techniques used in ContentMap. For additional information about the methods and proofs refer to [2].

3.1 Computation of the Mappings

Ontology mappings can be computed using one or more mapping tools (in [2] we used for our experiments OLA [3], AROMA[4] and CIDER[5]). ContentMap loads pre-computed mappings in the form of an OWL 2 ontology and provides a GUI for visualising the mappings (see Figure 2). Users can either manually accept or reject mappings or automatically filter them by setting a confidence threshold.

[3]OWL Lite Alignment: http://ola.gforge.inria.fr/

[4]AROMA: http://www.inrialpes.fr/exmo/people/jdavid/

[5]CIDER: http://sid.cps.unizar.es/SEMANTICWEB/ALIGNMENT/

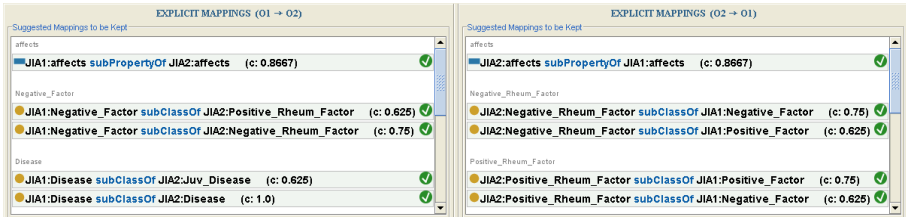


Figure 2. GUI for Visualising Explicit Mappings in ContentMap

3.2 Computation of New Entailments and Error Detection

To help users understand the semantic consequences of the integration, they should be informed about new entailments that hold in the merged ontology $\mathcal{U}$, but not in $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{M}$ alone. To this end, we use the notion of *logic difference* [4]. Intuitively, the logical difference between $\mathcal{O}$ and $\mathcal{O}'$ w.r.t a signature Σ is the set of entailments constructed over Σ that do not hold in $\mathcal{O}$, but do hold in $\mathcal{O}'$.

The notion of logic difference, however, has several drawbacks in practice. First, there is no algorithm for computing the logic difference in expressive DLs such as $\mathcal{SROIQ}$ (OWL 2) and $\mathcal{SHOIQ}$ (OWL DL) [4]. Second, the number of entailments in the difference can be huge (even infinite), and so likely to overwhelm users.

The GUI implemented in ContentMap allows users to customise approximations of the logic difference by selecting among the following simple kinds of entailment, where A, B are atomic concepts (including $\top, \bot$) and R, S atomic roles or inverses of atomic roles: *(i)* $A \sqsubseteq B$, *(ii)* $A \sqsubseteq \neg B$, *(iii)* $A \sqsubseteq \exists R.B$, *(iv)* $A \sqsubseteq \forall R.B$, and *v)* $R \sqsubseteq S$. The smallest implemented approximation considers only axioms of the form *(i)* (i.e. reasoner output), while the largest one considers all types *(i)—(v)*.

Figure 3 represents the ContentMap GUI where the entailments (for the selected logic difference approximation) are shown and can be selected to create $\Im^+$ (intended entailments to keep) and $\Im^-$ (unintended entailments to delete). ContentMap also provides a mechanism that automatically proposes candidate entailments to populate $\Im^+$ and $\Im^-$ sets.

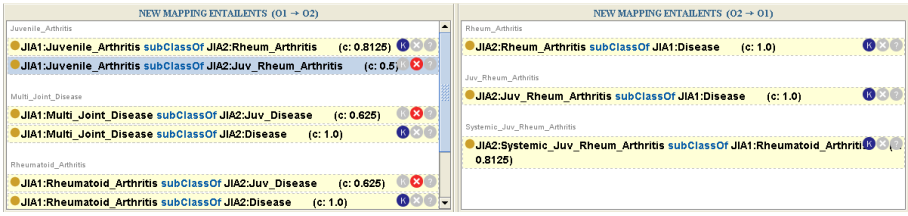


Figure 3. GUI for Visualising New Entailments in ContentMap

3.3 Computation of Repair Plans

If the user has selected one or more unintended entailments (i.e., $\Im^- \neq \emptyset$), then $\mathcal{U}$ clearly contains errors. Errors could be due to erroneous mappings, to inherently conflicting information in the two ontologies, or to some combination of both. Repairing such errors might, therefore, require the removal of axioms from one or more of $\mathcal{M}$, $\mathcal{O}_1$ and $\mathcal{O}_2$. Additionally, any removal of axioms should also respect $\Im^+$. A repair plan for $\mathcal{U}$ given $\Im^+$ and $\Im^-$ is a set $\mathcal{P} \subseteq \mathcal{U}$ such that: *1)* $(\mathcal{O} \setminus \mathcal{P}) \models \alpha$ for each $\alpha \in \Im^+$, and *2)* $(\mathcal{O} \setminus \mathcal{P}) \not\models \beta$ for each $\beta \in \Im^-$. Notice that, our approach is related to existing approaches for revising mappings [5] and for debugging and repairing inconsistencies in OWL ontologies [6].

Figure 4 shows an example of a set of extracted repair plans, for a given selection of $\Im^-$ and $\Im^+$. When displaying plans, the GUI indicates whether the axioms in the plan come from $\mathcal{O}_1$, $\mathcal{O}_2$, or $\mathcal{M}$ (marked with '1', '2' and 'M' respectively). Additionally axioms shared by all plans are marked with a 'P'.

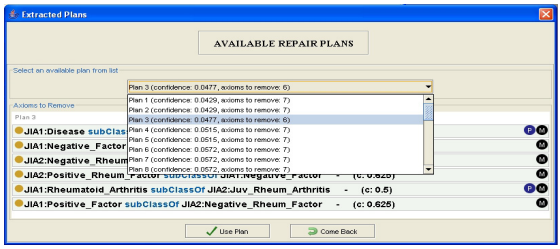


Figure 4. Selection of available Repair Plans in ContentMap

References

1. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag (2007)
2. Jimenez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Ontology integration using mappings: Towards getting the right logical consequences. In: Proc. of European Semantic Web Conference (ESWC). Volume 5554 of LNCS., Springer-Verlag (2009) 173–187 Technical Report and Tool also available at: `http://krono.act.uji.es/people/Ernesto/contentmap`.
3. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. J. Web Semantics **6**(4) (2008) 309–322
4. Konev, B., Walther, D., Wolter, F.: The logical difference problem for description logic terminologies. In: Proc. of IJCAR. (2008) 259–274
5. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Reasoning Support for Mapping Revision. Journal of Logic and Computation (2008)
6. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B.: Repairing unsatisfiable concepts in OWL ontologies. In: Proc. of ESWC. (2006) 170–184

SPARQL Query Splitter: query translation between different contexts ⋆

Carlos R. Osuna[1], David Ruiz[1], Rafael Corchuelo[1], and José L. Arjona[2]

[1] University of Sevilla
{carlosrivero,druiz,corchu}@us.es
[2] University of Huelva
jose.arjona@dti.uhu.es

Abstract. When integrating the information of a number of applications, one solution is to provide one or more unified entry points to the applications. A challenging task in this context is to translate the user query, which is posed over the unified entry point, to the integrated applications. The user query is expressed in terms of the entry point schema and the integrated applications do not accept this query because they have different schemes. In this demo, we present SPARQL Query Splitter, a tool that translates SPARQL user queries posed over an entry point schema to a number of queries over the application schemes. Our tool uses the mappings between the schemes to perform the translation by means of a symbolic calculator.

Key words: Enterprise Information/Application Integration, Web Databases, Semantic Web and Ontologies

1 Introduction

Enterprise Information Integration (EII) [2] consists of providing a unified entry point to a number of applications, which are integrated. The main feature of EII is that the information is retrieved on-line. In the bibliography, there are two main approaches to integrate applications using the EII paradigm: Mediators and Peer Data Management Systems (PDMS).

Mediators [3] offer a unique schema (i.e., data model) as the unique entry point to the integrated applications. PDMS [8] are the evolution of the Mediators and they offer a number of schemes to the integrated applications. In all EII solutions, there are a number of mappings that semantically relate the schemes used as entry points, to the schemes of the integrated applications. In the database research field, these mappings correspond to global-as-view (GAV), local-as-view (LAV) and hybrid approaches [9].

A key problem in this context is the translation of user queries, posed over the entry point schemes, to the application schemes. In the database research field, this problem is solved by using query rewriting/answering or unfolding techniques [6]. These techniques work with views and they are based on relational models, not taking into account other data models such as XML or those used in the Semantic Web arena. Some

⋆ This work has been partially supported by the Spanish and the Andalusian Governments under project IntegraWeb: TIN2007-64119, P07-TIC-2602 and P08-TIC-4100.

approaches adapt these techniques to work with the hierarchical model of XML or RDF [7]. Other approaches work with views on XML [10] or RDF [11] data models.

Other approaches are not based on views and they translate the user queries by applying some rules that indicate how to translate the user query into an application query. These rules can be specific for each pair of schemes [5] or more generic rules based on patterns [1]. A problem of these approaches is that the building and maintenance of the rules is costly. Another problem is that the rules can be contradictory or overlapping and, in these cases, the translation of the user query can not be performed.

In this demo, we present SPARQL Query Splitter[3], a tool for SPARQL query translation through different schemes or contexts. SPARQL Query Splitter is not based on views and it uses the mapping between the schemes to perform the translation. The tool uses the symbolic calculus to replace the attributes of the user query with the attributes of the application schemes. We present the implementation details of the tool in Section 2 and a motivating example in Section 3.

2 The Tool

In our tool, the schemes are RDFS ontologies that has a number of attributes. An attribute describes a path within the ontology, e.g., in Figure 2, the travel schema has an attribute of the form: 'Travel.cityFrom.hasAirport.airportName', and another attribute: 'Travel.cityTo.hasAirport.airportName'. We justify the use of paths because it is impossible to distinguish between which airport name is referenced in an attribute such as: 'Airport.airportname'.

1) Travel.hasDuration.begin = Booking.arrivalDate
2) Travel.hasDuration.end - Travel.hasDuration.begin = Booking.numberOfNights
3) Travel.cityFrom.hasAirport.airportName = Flight.airportFrom
4) Travel.cityTo.hasAirport.airportName = Flight.airportTo
5) Travel.cityFrom.cityTo = *citiesOfAirport*(Flight.airportFrom)
6) Travel.cityTo.cityTo = *citiesOfAirport*(Flight.airportTo)

Fig. 1. Mapping expressions between travel schema and hotel and flight schemes

There are a number of mapping expressions that relate the virtual schema (i.e., the unique entry point) to the application schemes. For example in Figure 1, mapping expressions 1 and 2 relate the travel schema to the hotel schema (cf. Figure 2), and mapping expressions 3, 4, 5 and 6 relate the travel schema to the flight schema.

Having these mapping expressions, a user query is posed over the virtual schema. The SPARQL Query Splitter works in three steps for each application schema: first, it isolates the attributes of the virtual schema in the mapping expressions; second, it replaces the isolated attributes in the user query, so the query is expressed in terms of

[3] The SPARQL Query Splitter demo is available on-line: `http://150.214.188.90:8080/SQPWeb/`

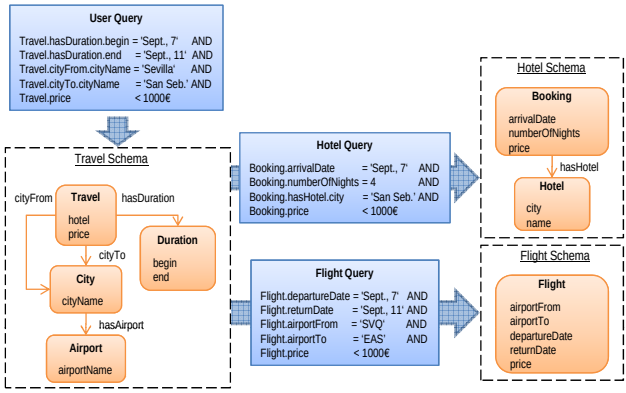


Fig. 2. A user query posed over the travel schema is divided into a query over the hotel schema and another over the flight schema

attributes of the current application schema; third, it tries to simplify the final query. For the three steps, we use a symbolic calculator program such as Mathematica or Maple [4]. The symbolic calculator, usually, does not work with SPARQL, so we transform the SPARQL query into a MathML [4] expression. The symbolic calculator make its operations and it responses with another MathML expression, we perform the inverse transformation to obtain the new SPARQL query.

1) Initial mapping expressions:
   Travel.hasDuration.begin = Booking.arrivalDate
   Travel.hasDuration.end - Travel.hasDuration.begin = Booking.numberOfNights
2) Isolation of travel attributes:
   Travel.hasDuration.begin = Booking.arrivalDate
   Travel.hasDuration.end = Booking.arrivalDate + Booking.numberOfNights
3) User query:
   Travel.hasDuration.begin = Sept, 7 AND
   Travel.hasDuration.end = Sept, 11
4) Replaced user query:
   Booking.arrivalDate = Sept, 7 AND
   Booking.arrivalDate + Booking.numberOfNights = Sept, 11
5) Simplify query:
   Booking.arrivalDate = Sept, 7 AND
   Booking.numberOfNights = 4

Fig. 3. An example of the process of the query translation

[4] http://www.wolfram.com/products/mathematica/
http://www.maplesoft.com/products/Maple/

3 The Demo

The example of our SPARQL Query Splitter tool is a travel booking scenario (cf. Figure 2), a user wishes to go from Sevilla to San Sebastián between September, 7 to September, 11 and the price of the travel must not exceed 1000 €. The user poses this query over the travel (virtual) schema. When translating the query to the hotel schema, the attributes of the travel schema are isolated in the mapping expressions as can be seen in the first and second step in Figure 3. The isolated mapping expressions are replaced into the user query and it corresponds to steps third and fourth in Figure 3. Finally, the last step consist of simplifying this query and the result is the fifth step in Figure 3.

The same process is performed to translate the user query from the travel schema to the flight schema. Note that the departure city, Sevilla, and the arrival city, San Sebastián, in the user query (cf. Figure 2) are transformed into the departure airport, SVQ, and the arrival airport, EAS, in the flight query.

References

1. S. Benbernou and M.-S. Hacid. Resolution and Constraint Propagation for Semantic Web Services Discovery. *Distributed and Parallel Databases*, 18(1), 2005.
2. P. A. Bernstein and L. M. Haas. Information integration in the enterprise. *Commun. ACM*, 51(9), 2008.
3. J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1), 2008.
4. D. Carlisle et al. Mathematical Markup Language (MathML) Version 2.0. Technical report, W3C, 2001.
5. K. C.-C. Chang and H. Garcia-Molina. Approximate query mapping: Accounting for translation closeness. *VLDB J.*, 10(2-3), 2001.
6. A. Y. Halevy et al. Answering queries using views: A survey. *VLDB J.*, 10(4), 2001.
7. A. Y. Halevy et al. Piazza: data management infrastructure for semantic web applications. In *WWW*, 2003.
8. A. Y. Halevy et al. Schema mediation for large-scale semantic data sharing. *VLDB J.*, 14(1), 2005.
9. M. Lenzerini et al. Data Integration: A Theoretical Perspective. In *PODS*, 2002.
10. N. Onose et al. Rewriting nested xml queries using nested views. In *SIGMOD Conference*, 2006.
11. B. Quilitz and U. Leser. Querying Distributed RDF Data Sources with SPARQL. In *ESWC*, 2008.

Annotator: Herramienta para la Anotación Semántica de Islas de Datos Amigables en la Web

José L. Álvarez, José L. Arjona, Agustín Domínguez, Nicolás Amador

Departamento de Tecnologías de la Información. Universidad de Huelva
E.P.S. La Rábida, Ctra. Huelva-La Rábida, 21071 Huelva
{alvarez, jose.arjona}@dti.uhu.es ; {agustin.dominguez, nicolas.amador}@alu.uhu.es

Introducción

La Web es el mayor repositorio de conocimiento de la Humanidad y ha conseguido un gran éxito gracias a que la mayor parte de la información que ofrece está disponible a través de mecanismos de interacción amigables y en un formato fácil de entender por las personas. Esta es la razón por la que muchos autores hacen referencia a los sitios Web como islas de datos amigables en la Web. Por otro lado, los Servicios Web y la Web Semántica están revolucionando este panorama; en cuanto están emergiendo nuevas tecnologías que, aportando contenido semántico además del sintáctico, facilitan el acceso a la Web a agentes software que ofrecen innumerables posibilidades para, entre otras, la integración de aplicaciones y de información en entornos Web.

El problema para utilizar estas tecnologías en sitios Web ya existentes recae en la necesidad de una reingeniería del sitio que no siempre es viable. Ante estas circunstancias, una buena solución consiste en utilizar algoritmos, denominados *wrappers*, capaces de obtener datos de interés desde un sitio Web. Estos algoritmos utilizan un conjunto de páginas de entrenamiento que, en muchos casos, requieren de un proceso de etiquetado o anotación de la información de interés en la página.

Por otro lado, analizar con técnicas de minería de datos las características de un sitio Web permite obtener modelos capaces de clasificar las páginas, obtener patrones repetitivos, etc.

La herramienta que presentamos en este trabajo, denominada *Annotator: Semantic Web Tool*, facilita el proceso de reingeniería ayudando, así, a los ingenieros a desarrollar soluciones de integración de islas de datos amigables en la Web. En concreto, ofrece la posibilidad de anotar semánticamente los contenidos de un sitio Web, da soporte para la construcción de extractores de información y permite el cálculo de características para construir clasificadores de páginas Web.

Actas JISBD 2009, pp.324-327, ISBN: 978-84-692-4211-7

Fig. 1. Interfaz principal de la herramienta *Annotator: Semantic Web Tool*

Annotator: Semantic Web Tool

La herramienta que proponemos, cuya interfaz de usuario se muestra en la figura 1, tiene como objetivo dar soporte al proceso de dotación de semántica para un sitio Web ya existente. Las principales tareas de esta herramienta, desarrolladas en los siguientes apartados, son anotar cada página Web a partir de una ontología (en formato OWL) almacenando el XPOINTER de cada anotación y el cálculo de características de cada página para crear un conjunto de datos para posteriormente ser analizado con técnicas de minería de datos.

Además de estas tareas, *Annotator* también ofrece un conjunto de utilidades, más simples, pero de gran ayuda para el análisis de un sitio Web concreto. Entre ellas:

- *Browser* para facilitar la navegación y el procesado de cada página.
- Visor del árbol DOM de la página procesada, con sincronización durante el marcado con el Browser.
- Visor de los atributos del elemento HTML procesado.
- Procesador de consultas XPATH para analizar el contenido de la página Web.
- Visor de la ontología cargada para la anotación y de los elementos anotados.

Anotación de Páginas Web

La anotación de páginas Web es, como se ha mencionado anteriormente, una tarea necesaria para utilizar un Extractor de Información Supervisado en un sitio Web. Annotator permite cargar una ontología OWL para anotar los elementos de interés de una página Web, obteniendo su XPOINTER, tanto a nivel de clase como de atributo.

Fig. 2. Proceso de anotación de una página a partir de una Ontología OWL

Una vez cargada la ontología, si marcamos un registro de la página Web como se aprecia en la figura 2, se podrá asociar éste a una clase concreta de la ontología y posteriormente se podrá ir asociando cada contenido del registro (título, autor, precio, etc.) a un atributo de esa clase. En la figura 3 se muestra la anotación de cada atributo.

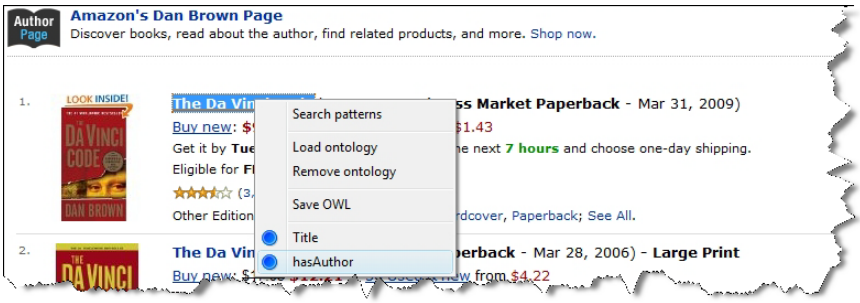


Fig. 3. Anotación de cada elemento de un registro asociándolo a un atributo de la clase

Cálculo de características para aplicar minería de datos

El segundo objetivos de esta herramienta es obtener un conjunto de características de cada página Web para posteriormente aplicar técnicas de minería de datos que permitan analizar un sitio Web.

Para alcanzar este objetivo, *Annotator* ofrece la posibilidad de generar un fichero en formato *.arff* que puede ser analizado mediante la herramienta *Weka* [3]. Por defecto, las características disponibles incluyen todas las etiquetas (*Tags*) HTML, resultados de consultas XPATH y dispone de un sistema de *plugins* que permite añadir todas las que el usuario estime oportunas.

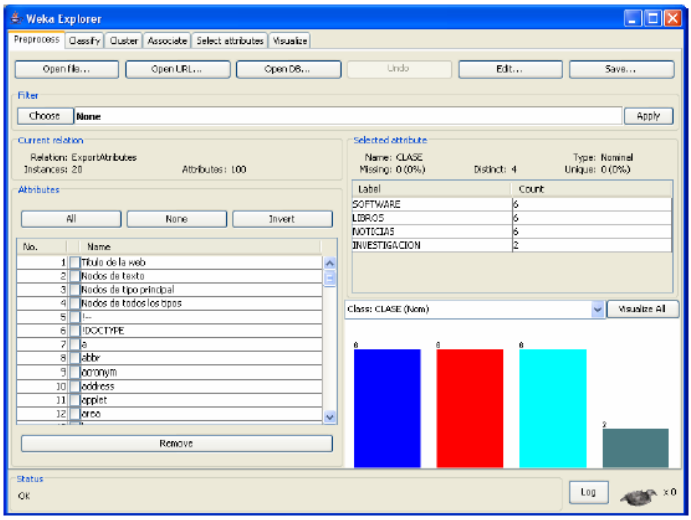


Fig. 4. Herramienta Weka analizando un sitio Web generado por Annotator

Conclusiones

Annotator: Semantic Web Tool [2] es una herramienta para dar soporte a la Integración de Aplicaciones Web ofreciendo ayuda para anotar las páginas de los sitios Web que se desean integrar y sobre los cuales será necesario aplicar algún extractor de información supervisado. Esta herramienta ha sido desarrollada dentro del proyecto *IntegraWeb* [1] (TIN2007-64119, P07-TIC-02602, P08-TIC-4100) y está siendo usada como base para los algoritmos de extracción de los proyectos.

Además de la anotación, esta herramienta permite la creación de un conjunto de datos con las características de las páginas Web de un determinado sitio para aplicar técnicas de minería de datos que permitan analizar el sitio Web, con WEKA [3]. Dentro del mismo proyecto, esta tarea está en fase de estudio para la clasificación de las páginas Web, de tal forma que se puedan distinguir entre páginas de datos, de índices, de error y sin interés, y también para la agrupación de nodos (del árbol DOM) que permitirán la identificación de estructuras comunes.

Entre los temas de interés de las jornadas con los que la herramienta *Annotator* presenta relación son los siguientes: Ingeniería Web; Minería de Datos y Almacenes de Datos; Recuperación de Información, Indexación y BD en Web; XML y Datos Semiestructurados; Web Semántica y Ontologías.

Referencias

1. IntegraWeb, http://www.tdg-seville.info/Projects/IntegraWeb/
2. Annotator, http://www.tdg-seville.info/Projects/IntegraWeb/ Annotator-1.3.zip
3. Weka Web Site, http://www.cs.waikato.ac.nz/ml/weka

Parte X

Sesión 9. Validación/Modelado Conceptual

Extensión UML para Casos de Uso Reutilizables en entornos Grid Móviles Seguros

David G. Rosado[1], Eduardo Fernández-Medina[1] y Javier López[2]

[1] UCLM. Grupo Alarcos – Instituto de Tecnologías y Sistemas de Información. Dep. de Tecnologías y Sistemas de Información – ESI, Paseo de la Universidad 4, 13071 Ciudad Real
{David.GRosado, Eduardo.FdezMedina}@uclm.es
[2] Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga, jlm@lcc.uma.es

Resumen. Los sistemas Grid nos permiten construir sistemas complejos con características diferenciadoras (interoperabilidad entre múltiples dominios de seguridad, autenticación y autorización a través de dominios, sistema dinámico y heterogéneo, etc.). Con el desarrollo de la tecnología wireless y los dispositivos móviles, el Grid llega a ser el candidato perfecto para que los usuarios móviles puedan realizar trabajos complejos, a la vez que añaden nueva capacidad computacional al Grid. Estamos construyendo un proceso completo de desarrollo para sistemas Grid móviles seguros, y una de las actividades es el análisis de requisitos, que está basado en casos de uso reutilizables. En este artículo, presentaremos una extensión UML para casos de uso de seguridad y Grid, los cuales capturan el comportamiento de este tipo de sistemas. Esta extensión UML está siendo aplicado a un caso real para construir diagramas de casos de uso de la aplicación, incorporando los aspectos de seguridad necesarios.

Palabras Claves: Seguridad, Casos de uso de Seguridad, desarrollo seguro, Grid Móvil seguro, Reutilización.

1 Introducción

La creciente necesidad de construir sistemas seguros, debido principalmente a las nuevas vulnerabilidades derivadas del uso de Internet y de las aplicaciones distribuidas en entornos heterogéneos, motiva a la comunidad científica a demandar una clara integración de la seguridad dentro de los procesos de desarrollo [1-4]. Un tipo de sistemas que tiene características diferenciadoras claras [5-7], y donde la seguridad es un factor de suma importancia, son los sistemas basados en Grid Computing. Los procesos de desarrollo genéricos son usados para desarrollar sistemas sin tener en cuenta ni el entorno tecnológico subyacente, ni las características y particularidades de estos sistemas específicos.

La seguridad es un aspecto central en la computación Grid desde el principio, y ha sido considerado como el cambio más significativo de la computación Grid [8, 9]. Además, la seguridad es más difícil de implementar dentro de una plataforma móvil

debido a las limitaciones de recursos de los dispositivos móviles [10]. Por tanto, una infraestructura Grid que soporte la participación de nodos móviles jugará un importante papel en el desarrollo de la computación Grid.

La mayoría de aplicaciones Grid existentes se han construido sin un proceso sistemático de desarrollo, basándose en desarrollos ad-hoc [5, 11]. La falta de métodos de desarrollo adecuados para este tipo de sistemas nos ha motivado a construir una metodología para desarrollarlos [12-14], ofreciendo una guía detallada para analizarlos, diseñarlos e implementarlos. La metodología está fuertemente orientada hacia la reutilización, y especialmente sensibilizada con la seguridad y la utilización de dispositivos móviles en los Grids Computacionales. La reutilización se concentra principalmente i) en la etapa de análisis en la que se parte de un conjunto de casos de uso predefinidos (y diagramas de interacción), y que se integran con los casos de uso identificados para una nueva aplicación y ii) en la etapa de diseño, en la que se parte de una arquitectura que incorpora los servicios de seguridad reutilizables previamente identificados, y se especializa para cada una de las nuevas aplicaciones que sean creadas.

En este artículo, presentaremos una extensión UML para casos de uso Grid reutilizables que pueda ser usada en la actividad de análisis, junto con diagramas de interacción y escenarios, para construir diagramas de casos de uso, integrando los requisitos para las aplicaciones Grid móviles seguras específicas. Esta extensión define los estereotipos necesarios para poder especificar detalles sobre los casos de uso Grid y casos de uso de seguridad que sirven de ayuda en la construcción de diagramas de casos de uso para este tipo de sistemas. También aplicamos esta extensión UML a un caso real donde estos casos de uso reutilizables, almacenados en el repositorio, son usados para construir un diagrama general para esta aplicación. Este caso real es un escenario en el dominio periodístico, dentro del proyecto europeo GREDIA [15], donde trabaja un grupo de investigación de la Universidad de Málaga, ofreciendo controles de seguridad para la infraestructura Grid móvil que le da soporte.

El resto del artículo se organiza como sigue: En la sección 2, presentaremos el trabajo relacionado. En la sección 3, se definirán los estereotipos y asociaciones para los casos de uso Grid y los describiremos formalmente. En la sección 4, aplicaremos los nuevos esterotipos para construir un diagrama de casos de uso en un caso real. Terminaremos proponiendo las conclusiones y algunas líneas de investigación sobre el trabajo futuro en la sección 5.

2 Antecedentes

La idea de desarrollar software mediante procesos de desarrollo sistemáticos para mejorar la calidad del software no es nueva [16-18]. Sin embargo, todavía hay muchos sistemas de información, tal como los Grid computing, que no son desarrollados mediante metodologías adaptadas a sus características más diferenciadoras [5]. De hecho, no hemos encontrado propuestas para el desarrollo sistemático de sistemas Grid, a pesar de la demanda de estos sistemas en la comunidad científica.

Por ejemplo, los autores en [19] presentan una metodología para la integración de la seguridad en los sistemas software. Esta metodología está basada en el Proceso Unificado [18] y es llamada Proceso Unificado Seguro (SUP – Secure Unified Process) . El problema es que sólo ofrece una solución a muy alto nivel sin ofrecer "mecanismos prácticos" (por ejemplo, artefactos de seguridad específicos de Grid, o una arquitectura de seguridad de referencia) que permita implementar su propuesta en un corto espacio de tiempo y con un mínimo esfuerzo. Otra propuesta [20, 21] se concentra en proporcionar semánticas formales en UML para integrar consideraciones de seguridad dentro del proceso de diseño software. La propuesta presenta UMLSec, que es una extensión de UML y permite expresar información relevante de seguridad. UMLSec y nuestra propuesta son compatibles, mientras que los modelos desde UMLSec pueden ser usados para especificar aspectos de seguridad generales de los sistemas, nuestra propuesta podría ser usada para especificar características de seguridad para entornos Grid.

Por otro lado, la actual arquitectura Grid no tienen en cuenta los entornos móviles debido a que los dispositivos móviles no han sido considerados como recursos de computación válidos o interfaces en la comunidad Grid. Es actualmente cuando se está prestando más atención a integrar estas dos tecnologías emergentes, computación Grid y computación móvil, como muestran algunos trabajos en [22-25], aunque ninguno elabora la forma de incorporar los dispositivos móviles en la actual arquitectura Grid. La metodología que proponemos considera por un lado, la incorporación de dispositivos móviles como un recurso más del Grid, y por otro lado, esta incorporación se realiza desde el principio del desarrollo sistemático, considerando todos los aspectos de seguridad y limitaciones de estos dispositivos.

3 Extensión de casos de uso para entornos Grid móviles seguros

La estructura de la metodología que estamos definiendo sigue el ciclo clásico, donde tenemos una etapa de planificación, una de desarrollo que incluye análisis, diseño y construcción, y finalmente, de una etapa de mantenimiento, sin embargo, está especialmente diseñada para este tipo de sistemas, con características tan particulares. Detalle sobre las actividades de la metodología pueden encontrarse en [12-14].

La actividad de análisis está centrada en casos de uso, donde se define el comportamiento, acciones e interacciones con los implicados en el sistema (actores), obteniendo una primera aproximación a las necesidades y requisitos (funcionales y no funcionales) del sistema a construir. Esta actividad se apoya en la reutilización de casos de uso Grid, almacenados en el repositorio de donde se obtienen casos de uso correctos que definen un comportamiento común del sistema Grid, y que son muy utilizados en la mayoría de diagramas de casos de uso que se construyen para diferentes sistemas Grid.

Para definir diagramas de casos de uso reutilizables específicos para los sistemas Grid móviles, necesitamos extender el metamodelo de UML 2.0 y definir un nuevo perfil con nuevos estereotipos. Un estereotipo es una extensión del vocabulario de UML que permite crear nuevos bloques de construcción derivados de los existentes, pero específicos a un dominio concreto, en nuestro caso al dominio Grid computing.

En esta sección, presentamos la extensión GridUCSec-Profile mediante la cual es posible representar características Grid móvil específicas y aspectos de seguridad para los diagramas de casos de uso, obteniendo como resultado, diagramas de casos de uso para entornos Grid móviles seguros y reutilizables. Esta extensión ha sido construida como un perfil UML, el cual es un mecanismo de extensibilidad que permite adaptar las metaclases de un modelo haciendo posible la incorporación de nuevos elementos en un dominio.

3.1 Extensión UML: GridUCSec-Profile

Para la representación de casos de uso Grid y casos de uso de seguridad, ha sido definido un conjunto de estereotipos, los cuales han sido agrupados en paquetes, *GridUCSec* y *TypesGridUCSec* que son parte de GridUCSec-Profile.

El paquete *GridUCSec* (ver Fig. 1) está compuesto de casos de uso Grid, casos de uso de seguridad, casos de mal uso, asociaciones de permiso, protección, amenaza y mitigación, junto con los actores involucrados, todos ellos necesarios para capturar el mayor número de aspectos de seguridad para sistemas Grid. Este paquete tiene 11 estereotipos: 4 especializan a *UseCase*, 2 especializan a *Actor*, y 5 especializan a *DirectedRelationship* y *NamedElement*. Los estereotipos que componen este paquete serán definidos en las siguientes subsecciones.

El paquete TypesGridUCSec (ver Fig. 1) define los tipos de datos para los valores etiquetados de los estereotipos de GridUCSec-Profile, como son el nivel de protección y riesgo, tipos de permiso, de requisito, de activos, de ataque, etc. Este paquete está compuesto de 9 estereotipos que especializan la clase *Enumeration*.

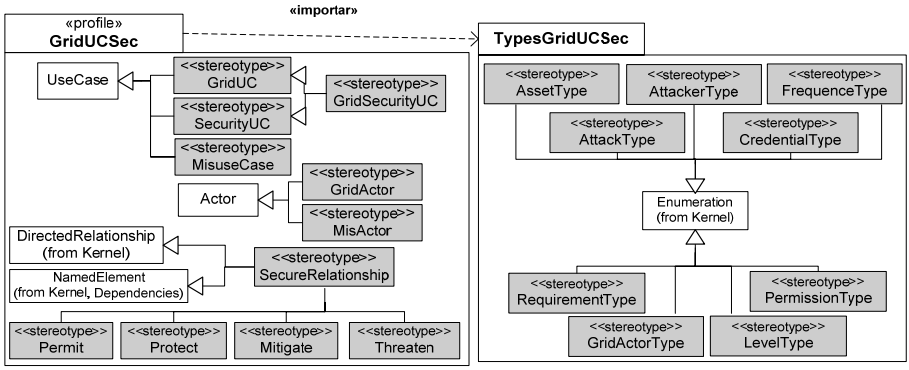


Fig. 1. Metamodelo de GridUCSec-Profile y TypesGridUCSec

En la Fig. 2, podemos ver las asociaciones entre estos nuevos estereotipos definidos para construir diagramas de casos de uso para sistemas Grid móviles seguros. Los estereotipos definidos para casos de uso se asocian con los estereotipos definidos para las relaciones entre casos de uso, indicando la acción (proteger, mitigar, permitir o amenazar) que ejerce un caso de uso sobre otro caso de uso mediante estos estereotipos de relación. Así, por ejemplo, el estereotipo «*SecurityUC*» se asocia con «*UseCase*» a través del estereotipo de relación «*Protect*» indicando que

el caso de uso de seguridad puede establecer una relación de protección con otro caso de uso. O dicho de otra forma, el estereotipo de relación «*Protect*» relaciona un caso de uso de seguridad con otro caso de uso para establecer la relación de protección. Hay varias restricciones que debemos definir y también, debemos describir en profundidad estas asociaciones de forma detallada para construir un perfil UML completo y correcto en el contexto de casos de uso para aplicaciones Grid. Esta descripción detallada será mostrada en la siguiente subsección.

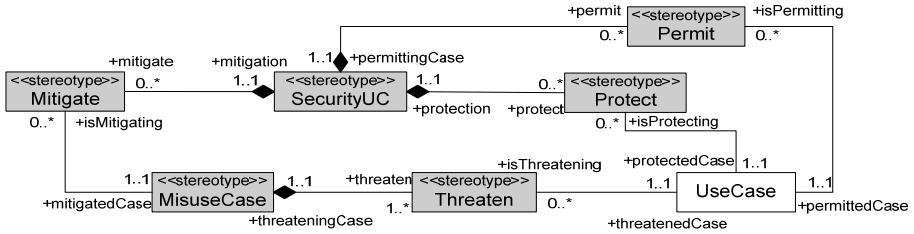


Fig. 2. Relación entre UseCase y DirectedRelationship.

3.2. Descripción detallada de estereotipos para el paquete GridUCSec

Una vez identificados los estereotipos y asociaciones entre ellos, ahora damos una descripción detallada de cada uno de ellos, describiendo los valores etiquetados y restricciones que definen su semántica. Usamos una plantilla, mostrada en la Tabla 1, basada en la especificación UML 2.0 [26], para describir formalmente los estereotipos anteriormente definidos. Esta plantilla simplifica, extiende y proporciona descripciones de cada uno de los estereotipos, facilitando su reutilización. Los elementos de esta plantilla son: *Descripción*: Indica el propósito y significado para los distintos usuarios del estereotipo. *Generalización*: Identifica el elemento del metamodelo de UML (clase base) desde la cual es heredado el estereotipo. *Asociaciones*: Se identifican las relaciones que tiene el estereotipo, tanto con otros estereotipos como con los elementos de UML. *Notación*: Corresponde a un icono que se asocia al estereotipo para su representación gráfica. *Valores Etiquetados*: Se identifican los atributos asociados al estereotipo. *Restricciones*: Corresponde a la descripción de un conjunto de limitaciones que tiene el estereotipo y la forma en que el estereotipo se relaciona con el resto de los estereotipos y con los elementos de UML. Estas restricciones se describen en forma textual y son definidas mediante expresiones OCL.

Tabla 1. Descripción detalla de los estereotipos para el paquete GridUCSec (Los números en la primera columna indican: 1) Descripción; 2) Generalización; 3) Asociaciones; 4) Notación; 5) Valores Etiquetados; y 6) Restricciones)

	«GridUC»	A	«SecurityUC»	B
1	Especifica los requisitos del sistema Grid y representa el comportamiento común, las funciones y las relaciones para esta clase de		Especifica los requisitos de seguridad del sistema, describiendo las tareas de seguridad que los usuarios deben ser capaces de desempeñar	

	sistemas.	por medio del sistema.
2	Classifier::BehavioredClassifier::UseCase	
3	-isPermitting:Permit[0..*]. Referencia la relación Permit que está permitiendo a este UC. -isProtecting:Protect[0..*]. Referencia la relación Protect que está protegiendo a este UC. -isThreatening:Threaten[0..*]. Referencia la relación Threaten que está amenazando a este UC.	-mitigate:Mitigate[0..*]. Referencia la relación Mitigate que pertenece a este UC de seguridad. -permit:Permit[0..*]. Referencia a la relación Permit que pertenece a este UC de seguridad. -protect:Protect[0..*]. Referencia a la relación Protect que pertenece a este UC de seguridad.
4		
5	GridRequirement, ProtectionLevel, SecurityDependence, InvolvedAsset	SecurityRequirement, InvolvedAsset, SecurityDegree
6	context GridUC inv: self.GridRequirement→size()=1 inv: self.InvolvedAsset→size()=1 inv: self.ProtectionLevel→size()=1 inv: self.SecurityDependence→size()=1 inv: self.MisActor→size()=0 inv: self.GridActor→size()>=0	context SecurityUC inv: self.Threaten→size()=0 inv: self.SecurityRequirement→size()=1 inv: self.InvolvedAsset→size()=1 inv: self.SecurityDegree→size()=1 inv: (self.mitigation→size() + self.permit→size()+ self.protect→size())>=1 inv: self.MisActor→size()=0 inv: self.GridActor→size()>=0
	«GridSecurityUC» **C**	**«MisuseCase»** **D**
1	Representan características de seguridad específicas de sistemas Grid. Especializa a los UC de seguridad comunes de otras aplicaciones, proporcionando características únicas para entornos Grid.	Representa una secuencia de acciones, incluyendo variantes, que un sistema u otra entidad puede desempeñar, interactuando con atacantes de la entidad y causando daño si la secuencia se completa.
2	- UseCase::SecurityUC; - UseCase::GridUC	- Classifier::BehavioredClassifier::UseCase
3	- Hereda asociaciones de SecurityUC. - Sólo hereda de GridUC la asociación isPermitting.	- threaten:Threaten [1..*]. Referencia la relación Threaten que pertenece a este caso de mal uso. - isMitigating:Mitigate [0..*]. Referencia la relación Mitigate que está mitigando este caso de mal uso.
4		
5	InvolvedAsset, SecurityRequirement, SecurityDegree, SecurityDependence	InvolvedAsset, ImpactLevel, RiskLevel, ThreatLikelihood, KindAttack
6	context GridSecurityUC inv: self.SecurityRequirement→size()=1 inv: self.InvolvedAsset→size()=1 inv: self.ProtectionLevel→size()=1 inv: self.SecurityDependence→size()=1. inv: self.Threaten→size()=0 inv: self.Protect→size()=0	context MisuseCase inv: self.Permit→size()=0 inv: self.Protect→size()=0 inv: self.MisActor→size()>=0 inv: self.GridActor→size()=0 inv: self.KindAttack→size()=1 inv: self.InvolvedAsset→size()=1 inv: self.ImpactLevel→size()=1 inv: self.RiskLevel→size()=1 inv: self.ThreatLikelihood→size()=1
	«Protect» **E**	**«Permit»** **F**
1	Esta relación especifica que el comportamiento de un UC podría ser protegido por el comportamiento de un UC de seguridad.	Esta relación especifica que el comportamiento de un UC podría ser permitido por el comportamiento de un UC de seguridad..
2	Element::Relationship::DirectedRelationship:: SecureRelationship Element::NamedElement:: SecureRelationship	
3	- protection:SecurityUC [1..1]. Referencia el UC que representa la protección y posee la relación protect.	- permittingCase:SecurityUC [1..1]. Referencia el UC que representa el permiso y posee la relación permit (SecurityUC o GridSecurityUC).

	- protectedCase:UseCase [1..1]. Referencia el UC que está siendo protegido.	- permittedCase: UseCase [1..1]. Referencia el UC que está siendo permitido (UseCase).
4	<<protect>>	<<permit>>
5	InvolvedAsset, ProtectionLevel, KindAttack	PermissionCondition, KindPermission
6	context Protect inv: (self.protectedCase→isTypeOf(UseCase) or self.protectedCase→isTypeOf(GridUC)) inv: (self.protection→isTypeOf(SecurityUC) or self.protection→isTypeOf(GridSecurityUC)) inv: self.InvolvedAsset→size()=1 inv: self.ProtectionLevel→size()=1 inv: self.KindAttack→size()=1 inv: self.InvolvedAsset= self.protectedCase.InvolvedAsset inv: self.IKindAttack= self.protectedCase.KindAttack	context Permit inv: (self.permitedCase→isTypeOf(GridUC) or self.permitedCase→ isTypeOf(GridSecurityUC)) inv: (self.permittingCase→ isTypeOf(SecurityUC) or self.permittingCase→ isTypeOf(GridSecurityUC)) inv: self.KindPermission→size()=1
	«Mitigate» G	**«Threaten» H**
1	Esta relación especifica que el comportamiento de un caso de mal uso podría ser mitigado por el comportamiento de un UC de seguridad.	Esta relación especifica que el comportamiento de un UC podría ser amenazado por el comportamiento de un caso de mal uso.
2	Element::Relationship::DirectedRelationship:: SecureRelationship Element::NamedElement:: SecureRelationship	
3	- mitigation:SecurityUC [1..1].]. Referencia el UC que representa la mitigación y posee la relación mitigate (SecurityUC o GridSecurityUC) - mitigatedCase: MisuseCase [1..1]. Referencia el UC que está siendo mitigado (MisuseCase).	- threatenedCase: UseCase [1..1]. Referencia el UC que está siendo amenazado (UseCase o GridUC). - threateningCase: MisuseCase [1..1]. Referencia el UC que representa la amenaza y posee la relación threaten.
4	<<mitigate>>	<<threaten>>
5	SuccessPercentage, KindCountermeasure	SuccessPercentage, KindVulnerability, KindAttack
6	context Mitigate inv: (self.mitigation→isTypeOf(SecurityUC) or self.mitigation→ isTypeOf(GridSecurityUC)) inv: self.SuccessPercentage→size()=1 inv: self.KindCountermeasure→size()=1	context Threaten inv: (self.threatenedCase→isTypeOf(UseCase) or self.threatenedCase→isTypeOf(GridUC)) inv: self.SuccessPercentage→size()=1 inv: self.KindVulnerability→size()=1 inv: self.KindAttack→size()=1
	«GridActor» I	**«MisActor» J**
1	Este actor especifica el rol jugado por un usuario Grid u otro sistema grid que interactúa con el sistema. **4** «GridActor»	Este actor especifica el rol jugado por un atacante o cualquier otro ataque que interactúa con el sistema. **4** «MisActor»
2	Classifier::Actor	Classifier::Actor
3	asociaciones con UseCase, GridUC, SecurityUC y GridSecurityUC	Sólo tiene asociaciones con MisuseCase
5	KindGridCredential, KindGridActor, KindRole, OrganizationName	KindMisActor, HarmDegree
6	context GridActor inv: self.KindGridCredential→size()>=0 inv: self.KindGridActor→size()=1 inv: self.KindRole→size()=1 inv: self.OrganizationName→size()>=1 inv: self.MisuseCase→size()=0	context Misuser inv: self.MisuseCase→size()>=1 inv: self.GridUC→size()=0 inv: self.SecurityUC→size()=0 inv: self.KindMisuser→size()=1 inv: self.HarmDegree→size()=1

3.3. Valores Etiquetados y estereotipos del paquete TypesGridUCSec

Presentamos en la Tabla 2, la descripción detallada de cada uno de los valores etiquetados definidos en el paquete GridUCSec. Para definir los valores etiquetados damos una pequeña descripción e identificamos los valores posibles de cada tipo asociados con el valor etiquetado (en la tabla se indica el número del tipo, que se puede ver en la Tabla 3).

Tabla 2. Valores Etiquetados de los estereotipos definidos en el paquete GridUCSec

Valor Etiquetado	Descripción	Nº Tipo
GridRequirement	Contiene los tipos de requisitos involucrados en el caso de uso.	9
HarmDegree	Define el grado de daño que un atacante puede causar al sistema.	7
ImpactLevel	Indica el nivel de impacto en el sistema que puede causar si alguna de las amenazas lleva a cabo un ataque con éxito.	7
InvolvedAsset	Identifica los activos que deben ser protegidos y que toman parte en la realización del caso de uso.	1
KindAttack	Describe el tipo de ataque que es llevado a cabo sobre el sistema abriendo un hueco de seguridad.	2
Kind Countermeasure	Describe las decisiones de cómo proteger la seguridad y privacidad del atacante potencial y de las vulnerabilidades.	10
KindGridActor	Define el tipo de actor Grid que interactúa con el sistema.	6
KindGrid Credential	Cuando un actor Grid quiere interactuar con el sistema, éste debe presentar o soportar un tipo de credencial de seguridad.	4
KindMisActor	El tipo de atacante que accede al sistema con el propósito de dañarlo.	3
KindPermission	Indica el tipo de permiso concedido para la realización de un UC.	8
KindRole	Indica el role de un actor en el sistema, así los privilegios asociados a un actor son conocidos.	10
Kind Vulnerability	Identifica los tipos de vulnerabilidades encontradas en el sistema y que son posibles candidatas de ataques.	10
OrganizationName	Todos los usuarios deben pertenecer a alguna organización.	10
Permission Condition	Define las condiciones necesarias bajo las cuales un caso de uso permite la realización de otro caso de uso.	10
ProtectionLevel	Indica el nivel de protección que el caso de uso debe tener.	7
RiskLevel	El nivel de riesgo es considerado en función de amenazas y vulnerabilidades de los activos.	7
SecurityDegree	Describe el grado de seguridad aportado por este UC al sistema.	7
Security Dependence	Indica si un caso de uso necesita ser asegurado porque su comportamiento puede generar un riesgo para el sistema.	7
Security Requirement	Contienen los tipos de requisitos de seguridad que intervienen en los casos de uso de seguridad.	9
Success Percentage	Indica el porcentaje de que una cierta acción (de seguridad, de ataque, etc.) tenga éxito.	7
ThreatLikelihood	Es la probabilidad que una amenaza sea llevada a cabo.	5

Para la definición de los estereotipos de la extensión GridUCSec, ha sido necesario definir tipos de datos, generales y específicos Grid, que representan los valores etiquetados y atributos de esos nuevos estereotipos (ver Tabla 3). Estos tipos están disponibles en el GridUCSec-Profile para que los atributos de los nuevos estereotipos puedan ser definidos y reutilizados.

Tabla 3. Valores de los Tipos definidos en el paquete TypesGridUCSec

Nº	Tipo	Valores
1	AssetType	Accounting, Credential, Data, General, Identity, Message, Resource, User.
2	AttackType	AccesControlAtt, ColludingAtt, DefeatingAtt, DoSAtt, EavesdroppingAtt, IntruderAtt, MaliciousAtt, MasqueradingAtt,ObjectReuseAtt, SniffingAtt
3	AttackerType	hacker, cracker, script kiddies, newbies, lamers, virus, Trojan, …
4	CredentialType	UserPass, X509, Kerberos, SAML, PIN, Biometric
5	FrequencyType	VFrequent, Frequent, Normal, Rare
6	GridActorType	User, MobileUser, Service, MobileResource, Resource, VO, Host
7	LevelType	VHigh, High, Medium, Low, VLow
8	PermissionType	Execute, CheckExecute, Interact, Include, Extend, Protect, Mitigate
9	Requirement Type	Accounting, Anonymity,Antivirus, Authentication, AuthenticationMutual, Authorization&AC, Availability, Confidentiality, Credential, Delegation, Firewall&IntrusionPrevention, Integration, Integrity, Interoperability, MappingIdentity, MultipleImplementation, NonRepudiation, Privacy, Revocation, Scalability, SSO, Trust, Usability
10	String	Cadena de caracteres

4 Ejemplo

La metodología que proponemos está siendo validada mediante su aplicación en un sistema real, una aplicación de negocio en el dominio multimedia y noticias, definido para el proyecto europeo GREDIA [15]. Queremos construir un sistema que ofrezca al periodista (en continuo movimiento) con un equipo ligero, la posibilidad de capturar y transmitir contenidos de información, manteniendo en todo momento los controles de seguridad necesarios.

Usando la extensión GridUCSec-Profile definida en este artículo, construiremos un diagrama de casos de uso para esta aplicación ayudándonos de los casos de uso reutilizables almacenados en el repositorio usado por la metodología propuesta. Por motivos de espacio y debido a su complejidad, sólo mostramos un caso de uso (Búsqueda Noticias) para este caso real de todos los definidos en el proyecto GREDIA (ver Fig. 3).

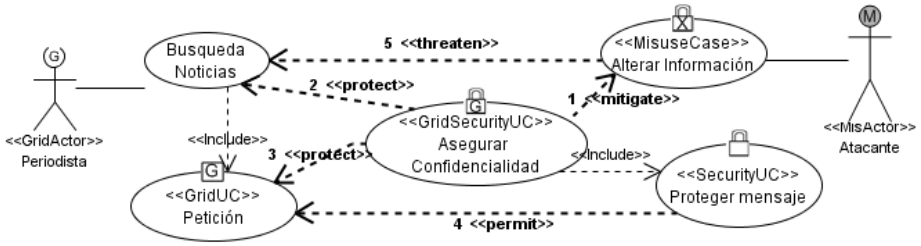


Fig. 3. Usando casos de uso reutilizables del repositorio para construir el diagrama final de la aplicación.

En este diagrama, el periodista («GridActor») hace una búsqueda de noticias (UseCase) a través del sistema para familiarizarse con el tema a tratar. Esta acción requiere hacer una petición al sistema Grid, representada por el caso de uso "Petición" («GridUC») que es reutilizable, al tratarse de un caso de uso común en sistemas Grid.

En este escenario, un atacante puede alterar la información que se trasmite a o desde el sistema Grid, por lo que es conveniente proteger los mensajes y el contenido de posible ataques. Todo esto se modela mediante un atacante («MisActor») que realiza el ataque de alterar la información («MisuseCase») sobre la búsqueda de noticias realizada por el periodista («GridActor»). Disponemos de un caso de uso de seguridad reutilizable que asegura la confidencialidad («GridSecurityUC») y mitiga el ataque, protegiendo la búsqueda de noticias y la petición en el Grid, a la vez que permite la realización de las peticiones, una vez haya sido protegido el mensaje. Este diagrama de casos de uso debe ser apoyado por diagramas de secuencia, que también estarán disponibles en el repositorio, para facilitar su comprensión y entender mejor las interacciones entre casos de uso y actores.

En la Tabla 4, mostramos la información relacionada con el diagrama de la

Fig. 3., describiendo los casos de uso y atributos siguiendo la extensión GridUCSec-Profile para este ejemplo. Todos los detalles que aparecen en la tabla deben ser definidos a la vez que construimos el diagrama, y son importantes para tomar decisiones de seguridad (definir servicios, mecanismos, etc.) en la actividad de diseño, para establecer la relación de trazabilidad entre casos de uso y servicios dentro de la arquitectura, o para el procesamiento automático en alguna herramienta CASE.

Tabla 4. Aplicación de GridUCSec-profile a un caso de estudio

	Estereotipos Asociaciones	**Asociaciones (Asoc) y Valores etiquetados (VEti) para los estereotipos de asociaciones**		
1	mitigate: Mitigate	Asoc	mitigation: Asegurar Confidencialidad mitigatedCase: Alterar información	
		VEti	KindCountermeasure: cifrar mensaje	SuccessPercentage: {High}
2	protect: Protect	Asoc	protection: Asegurar Confidencialidad	protectedCase: Búsqueda noticias
		VEti	InvolvedAsset: {Message} KindAttack: {MasqueradingAtt, EavesdroppingAtt}	ProtectionLevel: {High}
3	protect: Protect	Asoc	protection: Asegurar Confidencialidad	protectedCase: Petición
		VEti	InvolvedAsset: {Message} KindAttack: {MasqueradingAtt}	ProtectionLevel: {High}
4	permit: Permit	Asoc	permittingCase: Proteger mensaje	permittedCase: Petición
		VEti	PermissionCondition: todos los mensajes cifrados KindPermission: Execute	
5	threaten: Threaten	Asoc	threateningCase: Alterar información	threatenedCase: Búsqueda noticias
		VEti	SuccessPercentage: {High} KindVulnerability: mensajes por red inalámbrica KindAttack: {MasqueradingAtt, EavesdroppingAtt}	

Estereotipos	**Valores Etiquetados**	
«GridSecurityUC» Asegurar Confidencialidad	InvolvedAsset: {Message} SecurityDegree: {High}	Securityrequirement: {Confidentiality} SecurityDependence: {VLow}
«SecurityUC» Proteger mensaje	InvolvedAsset: {Message} SecurityDegree: {High}	SecurityRequirement: {Confidentiality}
«GridUC» Petición	InvolvedAsset: {Message} ProtectionLevel: {Medium}	GridRequirement: {Interoperatibility} SecurityDependence: {Medium}
«MisuseCase» Alteración información	ImpactLevel: {High} RiskLevel: {High}	InvolvedAsset:{Message,Identity,Data} KindAttack: {MasqueradingAtt} ThreatLikelihood: {Frequent}
«GridActor» Periodista	KindRole: periodista OrganizationName: News	KindGridActor: {Mobile User} KindGridCredential: {UserPass}
«MisActor» Atacante	KindMisActor: hacker	HarmDegree: {Medium}

5 Conclusiones y trabajo futuro

Una metodología de desarrollo para sistemas Grid móviles seguros nos sirve de guía para obtener un software de calidad, ofreciendo los métodos, técnicas y herramientas que faciliten la labor a todo el equipo involucrado en el desarrollo del software. Estudiando las necesidades y particularidades de los sistemas Grid móviles, fue necesario definir una extensión UML para casos de uso que capture el comportamiento, las funciones, las propiedades y necesidades que surgen en este tipo de sistemas. Esta extensión enriquece los diagramas de casos de uso con aspectos de seguridad y define valores que son esenciales a la hora de interpretar el diagrama de casos de uso en las sucesivas actividades del proceso de desarrollo.

La reutilización de elementos es una característica fundamental del proceso y ofrece a los desarrolladores e implicados en el desarrollo, soluciones construidas y probadas, mejorando la calidad del producto final, ahorrando tiempo y esfuerzo y mejorando la productividad en el proceso de desarrollo. Gracias a la extensión UML para casos de uso, podemos analizar los requisitos de seguridad del sistema desde las primeras etapas de desarrollo, y capturar toda la información de seguridad relevante que será necesaria en las siguientes actividades del proceso de desarrollo.

Como trabajo futuro se pretende completar el detalle de la metodología (actividades, tareas, etc.) a través del método de investigación-acción, integrar técnicas de ingeniería de requisitos de seguridad (UMLSec, etc.), y definir la trazabilidad de artefactos entre actividades.

Acknowledgments. Esta investigación es parte de los siguientes proyectos: QUASIMODO (PAC08-0157-0668) financiado por la "Viceconsejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha" (España), y ESFINGE (TIN2006-15175-C05-05) concedida por la "Dirección General de Investigación del Ministerio de Educación y Ciencia" (España). Agradecimiento especial a GREDIA (FP6-IST-034363) financiado por la Comisión Europea.

References

1. Bass, L., Bachmann, F., Ellison, R.J., Moore, A.P., Klein, M.: Security and survivability reasoning frameworks and architectural design tactics. SEI (2004)
2. Jürjens, J.: Secure Systems Development with UML. Springer-Verlag (2004)
3. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. 5th International Conference on the Unified Modeling Language (UML), 2002, Vol. 2460. Springer, Dresden, Germany (2002) 426--441
4. Mouratidis, H., Giorgini, P.: Integrating Security and Software Engineering: Advances and Future Vision. IGI Global (2006)
5. Kolonay, R., Sobolewski, M.: Grid Interactive Service-oriented Programming Environment. Concurrent Engineering: The Worldwide Engineering Grid. Press and Springer Verlag, Tsinghua, China (2004) 97–102
6. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: Grid services for distributed system integration. Computer 35 (2002) 37-46
7. Foster, I., Kesselman, C.: Globus: A Toolkit-Based Grid Architecture. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999) 259-278

8. Humphrey, M., Thompson, M.R., Jackson, K.R.: Security for Grids. Lawrence Berkeley National Laboratory. Paper LBNL-54853 (2005)
9. Chakrabarti, A., Damodaran, A., Sengupta, S.: Grid Computing Security: A Taxonomy. IEEE Security & Privacy 6 (2008) 44-51
10. Bradford, P.G., Grizzell, B.M., Jay, G.T., Jenkins, J.T.: Cap. 4. Pragmatic Security for Constrained Wireless Networks. In: Publications, A. (ed.): Security in Distributed, Grid, Mobile, and Pervasive Computing, University of Alabama, USA (2007) 440
11. Dail, H., Sievert, O., Berman, F., Casanova, H., YarKhan, A., Vadhiyar, S., Dongarra, J., Liu, C., Yang, L., Angulo, D., Foster, I.: Scheduling In The Grid Application Development Software Project. Grid resource management:state of the art and future trends (2004) 73-98
12. Rosado, D.G., Fernández-Medina, E., López, J., Piattini, M.: PSecGCM: Process for the development of Secure Grid Computing based Systems with Mobile devices. International Conference on Availability, Reliability and Security (ARES'08). IEEE, Barcelona, Spain (2008) 136-142
13. Rosado, D.G., Fernández-Medina, E., López, J., Piattini, M.: Engineering Process Based On Grid Use Cases For Mobile Grid Systems. The Third International Conference on Software and Data Technologies- ICSOFT 2008, Porto, Portugal (2008) 146-151
14. Rosado, D.G., Fernández-Medina, E., López, J.: Obtaining Security Requirements for a Mobile Grid System. International Journal of Grid and High Performance Computing (2009) (to be published in April 1, 2009)
15. GREDIA, www.gredia.eu
16. Bell, D.E., LaPadula, L.J.: Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997. Bedford, MA (1976)
17. Baskerville, R.: Information systems security design methods: implications for information systems development. ACM Computing Surveys 25 (1993) 375 - 414
18. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley Professional (1999)
19. Steel, C., Nagappan, R., Lai, R.: Chapter 8. The Alchemy of Security Design Methodology, Patterns, and Reality Checks. Core Security Patterns:Best Practices and Strategies for J2EE, Web Services, and Identity Management. Prentice Hall (2005) 1088
20. Jurjens, J.: Towards Development of Secure Systems Using UMLsec. In: LNCS, S.-V. (ed.): Fundamental Approaches to Software Engineering (FASE/ETAPS) (2001)
21. Jurjens, J.: UMLsec: Extending UML for Secure Systems Development. In: Springer (ed.): 5th International Conference on the Unified Modeling Language (UML), Vol. 2460, Dresden, Germany (2002) 1-9
22. Hans A. Franke, Fernando L. Koch, Carlos O. Rolim, Carlos B. Westphall, Douglas O. Balen: Grid-M: Middleware to Integrate Mobile Devices, Sensors and Grid Computing. Third International Conference on Wireless and Mobile Communications (ICWMC'07) Guadeloupe, French Caribbean (2007) 19
23. Isaiadis, S., Getov, V.: Integrating Mobile Devices into the Grid: Design Considerations and Evaluation. In: Jose C. and Medeiros, P.D. (ed.): 11th International Euro-Par Conference (Euro-Par 2005). LNCS (3648), Lisbon, Portugal (2005) 1080-1088
24. Jameel, H., Kalim, U., Sajjad, A., Lee, S., Jeon, T.: Mobile-To-Grid Middleware: Bridging the gap between mobile and Grid environments. European Grid Conference EGC 2005, Vol. 3470/2005. Springer, Amsterdam, The Netherlands (2005) 932-941
25. Phan, T., Huang, L., Dulan, C.: Challenge: Integrating Mobile Wireless Devices Into the Computational Grid. 8th annual international conference on Mobile computing and networking (MobiCom'02). ACM Press, Atlanta, Georgia, USA (2002) 271 - 278
26. OMG: OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2. (2007)

Perfiles UML en el diseño de notaciones visuales*

Jesús Pardillo and Cristina Cachero

Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Alicante, España
{jesuspv,ccachero}@dlsi.ua.es

Resumen La definición de perfiles en UML tiene algunas carencias reconocidas, entre las que destaca el fuerte acoplamiento existente entre semántica y notación a la hora de realizar la extensión. Este hecho, unido a las propias limitaciones que una aproximación icónica presenta a la hora de definir metáforas visuales efectivas, hace que muchos diseñadores se decanten por la definición de metamodelos, aún a costa de perder las ventajas de interconectividad que supone el mantenerse dentro del estándar. Con el fin de paliar esta situación, en este artículo presentamos la idea de enriquecer el metamodelo de perfiles UML con el soporte para extensiones notacionales complejas. A modo ilustrativo, tal enriquecimiento se ha diseñado mediante el metamodelo *Diagram Interchange*, que proporciona los constructores necesarios para soportar una extensión notacional compatible con UML. De este modo se aumenta la capacidad notacional de los perfiles al mismo tiempo que se separa el proceso de extensión semántica del proceso de extensión notacional.

Key words: UML, modelado, perfiles

1. Introducción a los perfiles UML

Los perfiles UML [18] son una técnica sencilla para la extensión de UML. Su uso permite que los diseñadores utilicen los elementos de modelado de UML para representar conceptos particulares a sus dominios, en lugar de tener que definir nuevos lenguajes de modelado desde cero. Para ello, los perfiles contienen *estereotipos* [21] que, aplicados sobre las *metaclases* de UML, permiten redefinir tanto su notación como su sintaxis y su semántica [2].

Con respecto a la notación, cada estereotipo puede llevar asociada una colección de iconos como medio para extender la notación de su metaclase base. Estos estereotipos son manejados por las herramientas de modelado de acuerdo con las reglas proporcionadas en UML ([18], pág. 690): por ejemplo, cuando la metaclase base se representa gráficamente como una caja, ésta puede sustituirse por el icono siempre y cuando se aplique un único estereotipo y las propiedades de la ocurrencia de la metaclase no se visualicen.

* Este trabajo ha sido financiado por los proyectos ESPIA (TIN2007- 67078) del Ministerio de Ciencia e Innovación y QUASIMODO (PAC08-0157-0668) del Ministerio de Educación y Ciencia de Castilla-La Mancha. El trabajo de Jesús Pardillo lo financia la beca FPU AP2006-00332 del Ministerio de Ciencia e Innovación.

Con respecto a la sintaxis, cada estereotipo puede venir acompañado de un conjunto de valores etiquetados que añaden atributos a la metaclase original.

Por último, con respecto a la semántica, ésta, al igual que ocurre en el propio UML, se suele redefinir de manera informal, mediante la asociación al estereotipo de una descripción en lenguaje natural [5]. A menudo, las extensiones semánticas imponen restricciones estructurales adicionales sobre la metaclase base. Tales restricciones se asocian a los estereotipos mediante lenguajes como OCL [17].

Por otro lado, en función del foco de extensión y de su nivel de complejidad, un estereotipo se puede clasificar en: *decorativo* (extensión notacional), descriptivo (extensión sintáctica), *restrictivo* (descriptivo con restricción semántica) y *redefinidor* (modificación de la semántica original de UML) [2]. A los perfiles que contienen exclusivamente estereotipos decorativos, descriptivos y/o restrictivos se les denomina *extensiones conservadoras* de UML, mientras que a los perfiles que contienen estereotipos redefinidores se les denomina extensiones *no* conservadoras de UML [20]. Si queremos que la semántica del elemento extendido no cambie la semántica de la metaclase base, la definición del estereotipos debe preservar una variante del principio de sustitución de Liskov [14], *i.e.*, debe ser posible utilizar cualquier instancia de una metaclase estereotipada en el lugar de las instancias de la metaclase sin estereotipar sin que cambie su semántica.

Aunque parece lógico pensar que los estereotipos siempre deberían intentar mantenerse dentro de los límites de la *no redefinición* semántica, en la práctica nos encontramos con que, a menudo, motivos meramente notacionales influyen en la selección de determinadas metaclases base de extensión, en detrimento de otras que, *a priori*, parecerían más apropiadas [13,22]. Es más, muchas veces ni siquiera con esta selección subóptima de metaclases base de la extensión es posible representar convenientemente los modelos perfilados, dada su compleja notación. Estas limitaciones notacionales han forzado a las herramientas de modelado a incorporar sus propias alternativas notacionales a las "reglas de extensión mediante iconos" de UML ([18], pág. 690). Por ejemplo, herramientas como la galardona MagicDraw [1] permiten definir la notación de las *asociaciones* de UML cambiando su forma y color o decorando sus extremos, esto es, realizando variaciones no normativas.

Esta situación puede ser parcialmente atribuida al acoplamiento entre notación, sintaxis y semántica en perfiles UML. Es por ello que este estudio se centra en los problemas que, durante el diseño de perfiles UML, derivan de este acoplamiento (§2), y cómo es posible solucionarlos considerando un metamodelo de perfilado UML enriquecido, que dé soporte a la definición ortogonal de notaciones (§3). Esta discusión concluye presentando las ventajas de esta aproximación (§4) a la hora de aumentar el atractivo de definición y uso de perfiles.

2. Acoplamiento semántico-notacional en perfiles UML

Como ya hemos comentado, uno de los problemas de definición de perfiles UML es su falta de potencia a la hora de soportar notaciones asociadas a los elementos extendi-

[1] www.magicdraw.com

dos. Este hecho causa en muchas ocasiones que el diseñador se encuentre con la disyuntiva de, a la hora de extender, tener que elegir entre hacerlo de la clase semánticamente más próxima al concepto que quiere modelar o hacerlo de la clase notacionalmente más parecida a la apariencia que quiere darle al concepto.

Si el diseñador selecciona una metaclase como base de extensión por lo adecuado de su notación, normalmente deberá pagar el precio de tener que modificar su sintaxis y/o semántica, complicando el proceso de definición de restricciones asociadas a los estereotipos. Un ejemplo lo tenemos en [15]. En este caso, (1) la necesidad de integración de un modelo de requisitos de i^*[2] en una herramienta que soportaba un proceso de desarrollo dirigido por modelos [10] basado en UML, y (2) la familiaridad de los usuarios finales de la herramienta con los diagramas de clases, provocó que la definición de este perfil modele *objetivos*, no instanciables, como clases, que sí lo son.

Si, por el contrario, el diseñador es riguroso y selecciona una metaclase por su semántica, deberá asumir su notación (que sólo se puede iconizar), independientemente de su idoneidad para representar visualmente el concepto en el lenguaje destino. Un efecto de esta situación se observa *e.g.* en muchos de los perfiles que extienden de los conceptos del diagrama de clases UML para adaptar conceptos en sus dominios que involucran estructuras de datos (ver *e.g.* [16]). En principio, las metaclases `Class`, `Association` y `Property` tienen una semántica idónea para ser utilizadas como base de estas extensiones. Sin embargo, con el aumento del número de clases y sus interrelaciones, la visualización en forma de diagrama de clases adolece de problemas de comprensibilidad asociados a su escalabilidad [8]. Si el lenguaje objeto es inherentemente complejo o debe visualizarse a gran escala, el diseño de una notación comprensible requiere recurrir a factores externos a UML, como la disposición visual de los elementos de modelado (*layout*) [9,7] o notaciones matriciales [11]. Sin embargo, el perfilado de UML no permite esta alternativa, ofreciendo como única posibilidad la asociación de iconos que, en este y otros muchos casos, no resuelve el problema notacional. Es más, incluso sin los problemas de escalabilidad, el éxito de la notación depende a menudo de factores asociados al usuario, como la comodidad o la estética [12], que no pueden manipularse con los perfiles UML.

En resumen, las reglas de diseño de notaciones mediante extensión de iconos de UML no parecen ser lo suficientemente versátiles ante las nuevas demandas de modelado, ni en cuanto a la mecánica de extensión ni en cuanto al objeto de la misma. Por un lado, la notación del modelo UML perfilado está limitada por tales reglas de diseño ([18], pág. 690), orientadas a la gestión de iconos y sin opción de manipulación significativa de la notación de UML. Por otro lado, las propias primitivas gráficas de UML no se ajustan siempre a las necesidades de visualización que se plantean en la práctica los diseñadores.

Para clarificar nuestra respuesta a este problema, a continuación presentamos un caso real que se nos ha planteado en el entorno académico.

[2] `istar.rwth-aachen.de`

Caso de estudio: diagramas ER en UML

En un intento de unificar herramientas y metodologías de desarrollo en las asignaturas de ingeniería del software y bases de datos, hace unos meses nos planteamos la posibilidad de definir un perfil UML para la notación del modelo entidad-relación (ER). El modelo ER es uno de los modelos más conocidos para representar bases de datos a nivel conceptual. Formula una representación del espacio de información en base a la dicotomía que le da nombre: un modelo ER clasifica principalmente los datos en `entidades`, `atributos` de diferentes tipos y `relaciones` entre ellos. Con su difusión, se han desarrollado variantes tanto en su capacidad expresiva (extensiones semánticas) como notacional. Tómese como ejemplo la Fig. 1 donde se muestran algunas de esas notaciones para representar las cardinalidades máximas en las relaciones.

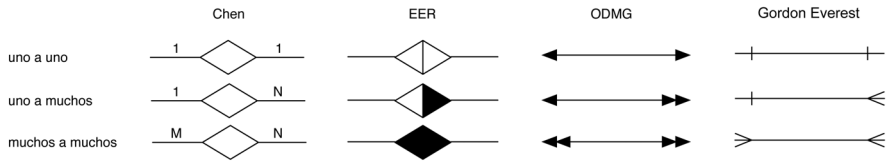


Figura 1. Ejemplos de variantes notacionales para las cardinalidades máximas del ER

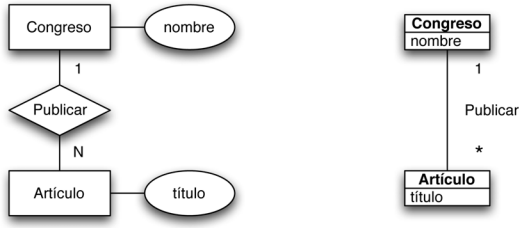


Figura 2. Diagrama ER (según Chen [4]) y diagrama de clases UML equivalente

Los elementos de modelado del ER pueden correlacionarse con metaclases de UML (ver Fig. 2). Concretamente, el concepto `entidad` en ER se expresa sin pérdida semántica como `Class` de UML, el de `relación` como `Association`, y el de `atributo` como `Property`. Esta capacidad expresiva del diagrama de clases permite asimismo representar elementos adicionales como cardinalidades o restricciones de existencia que son comunes en ambos lenguajes. Sin embargo, la extensión de estas metaclases de UML asume que los atributos (`nombre` y `título`) están contenidos visualmente en la primitiva gráfica de las entidades (`Congreso` y `Artículo`, respectivamente). Tal notación difiere de manera significativa de la notación ER tradicional, donde las entidades se visualizan como rectángulos y los atributos como elipses que se relacionan

gráficamente mediante líneas continuas. En nuestro caso, esto era un problema debido a que necesitábamos hacer transparente al alumno el hecho de estar usando el perfil en lugar de su herramienta habitual de bases de datos. Es por ello que optamos por definir algunos estereotipos a partir de metaclases que no eran las más adecuadas semánticamente. Como hemos comentado anteriormente, esta práctica de extender las metaclases de UML que mejor representan la notación visual pretendida, independientemente de la semántica, es una práctica bastante común. El perfil `ER--` de la Fig. 3 muestra nuestra solución de compromiso en este caso, que evitaba que los alumnos tuviesen que familiarizarse con dos notaciones distintas.

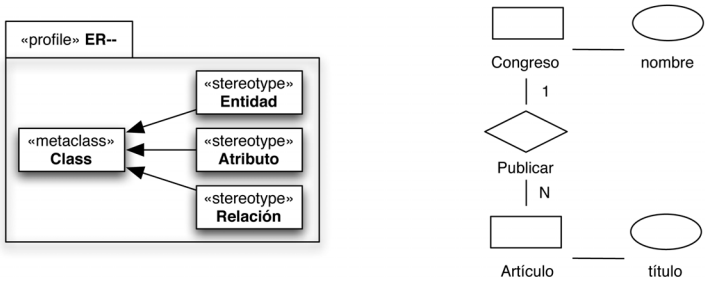


Figura 3. Perfil UML para diagramas ER y su aplicación al ejemplo de la Fig. 2

El perfil `ER--` define tres estereotipos para representar los elementos de modelado básicos del ER. Los tres estereotipos están definidos sobre la metaclase `Class` y por ello comparten su semántica. Respecto a su notación, en la Fig. 3 se puede observar cómo se han asociado los iconos ▭, ⬭ y ◇ para poder representar los diagramas de clases estereotipados de forma similar a los diagramas ER. Dado este perfil, el diagrama ER de nuestro caso de estudio (Fig. 2) puede representarse mediante el perfil UML como se observa en la Fig. 3.

La parte izquierda de la Fig. 4 muestra el diagrama de objetos correspondiente al diagrama ER de la Fig. 3. En esta figura se puede observar la complejidad que implica tal extensión durante el modelado: las entidades (*i.e.*, `Congreso` y `Artículo`), que han sido traducidas a clases de UML, deben enlazarse mediante secuencias de `(Property, Association, Property)` con sus atributos (*i.e.*, `nombre` y `título`) así como con su relación (`Publicar`). Por otro lado, la parte derecha de la Fig. 3 muestra el enlace de estos elementos de haberse respetado la semántica de las metaclases de UML: cada `Entidad`, `Atributo` y `Relación` se traduce a una metaclase UML más afín (`Class`, `Property` y `Association`). Como resultado, se eliminan los elementos de modelado artificiales que se debían introducir en la traducción con semántica alterada.

A la complejidad anterior se le suma la sobrecarga de restricciones necesarias para restringir la semántica, situación habitual cuando la metaclase base se extiende por motivo notacionales. Estas restricciones deben asegurar que los modelos realizados utilizando el perfil están bien formados según la sintaxis y semántica ER. Aunque una dis-

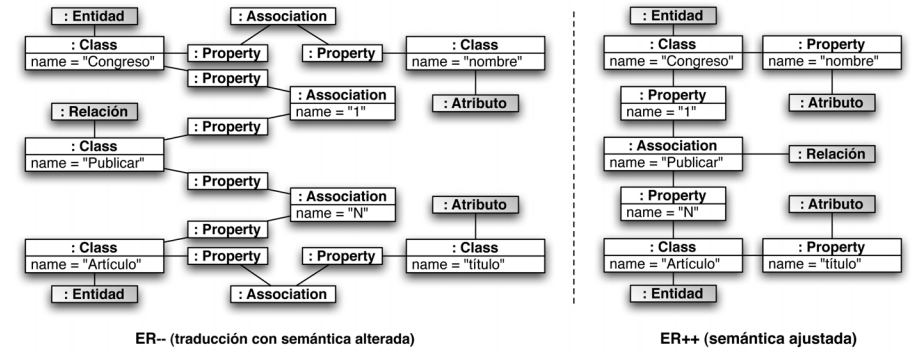


Figura 4. Diagrama de objetos del modelo UML perfilado (estereotipos sombreados)

cusión detallada de este perfil queda fuera del ámbito de este trabajo, algunos ejemplos de tales restricciones, expresadas formalmente en OCL, se presentan a continuación [3]. Estas restricciones permiten asegurar que los modelos realizados utilizando el perfil están bien formados según la sintaxis y semántica ER.

```
context Entidad
inv 'no features': self.base_Class.feature->isEmpty()
inv 'disjunta': self.base_Class.extension_Atributo->isEmpty()
   and self.base_Class.extension_Relación->isEmpty()
inv 'asociatividad': self.getAssociations()->forAll(a |
   a.memberEnd.type->exists(t |
      t.isStereotypedBy(Atributo)
         or t.isStereotypedBy(Relación)))
```

Estas restricciones ejemplifican las repercusiones, en términos de complejidad del perfil, de la extensión de metaclases cuando existen diferencias semánticas significativas entre la metaclase y el estereotipo. Mientras que una extensión de `Class`, `Property` y `Association` habría requerido un pequeño conjunto de restricciones sencillas, la extensión seleccionada (donde todos los estereotipos extienden de `Class`) hace que sea necesario especificar restricciones como *e.g.* `'no features'`, cuyo objetivo es evitar que se puedan definir atributos dentro de una clase estereotipada como «`entidad`». En general, dada la expresividad de UML [19], se hace habitual que un largo número de restricciones de índole similar deban especificarse para las metaclases con alta conectividad con el resto del metamodelo (como es el caso de las involucradas en este ejemplo: `Class`, `Property` y `Association`). Otro ejemplo de esta situación es la necesidad de validación de las asociaciones entre entidades y sus atributos, tales como la existente entre el `Congreso` y su `nombre` (ver Fig. 3 & 4), donde la semántica, nuevamente alejada de la semántica original de la metaclase `Association`, exige suprimir etiquetas de nombres, roles y cardinalidades.

[3] `isStereotypedBy` y `getAssociations` son operaciones de conveniencia, cuya definición puede consultarse en [22].

A continuación presentamos una alternativa a este modo de perfilar sin perder la flexibilidad notacional que requeríamos en nuestro caso de estudio.

3. Perfilado de la notación de UML

Con objeto de (i) facilitar la elección de metaclases base por motivos meramente semánticos y (ii) permitir el diseño de notaciones más potentes que las obtenidas como resultado de asociar iconos a los estereotipos, en esta sección se presenta una propuesta de extensión de la especificación de UML que permite la extensión independiente de notación y semántica.

Al igual que UML, sus perfiles están definidos mediante un metamodelo (que forma parte del propio UML). Sin embargo, dado que su objetivo es modelar extensiones de UML, el metamodelo de los perfiles UML puede interpretarse como un metametamodelo. Teniendo esto en mente, en la parte izquierda de la Fig. 5, se muestra un extracto de la especificación actual del metamodelo de perfiles UML. Se define la (meta-) metaclase `Stereotype` como elemento central de extensión. Esta metaclase hereda la semántica de `Class`, y por ello, tiene `Properties` (denominadas definiciones etiquetadas) y `Constraints` (a través de `Namespace::ownedRule`). `Stereotype` se asocia a metaclases base mediante relaciones de `Extension` (no mostradas en la Fig. 5). Respecto a la extensión notacional, este metamodelo permite asignar iconos (metaclase `Image`) a los estereotipos conforme a UML ([18], pág. 690). Esta última parte de los perfiles UML podría considerarse como el *metamodelo estándar de extensión notacional*.

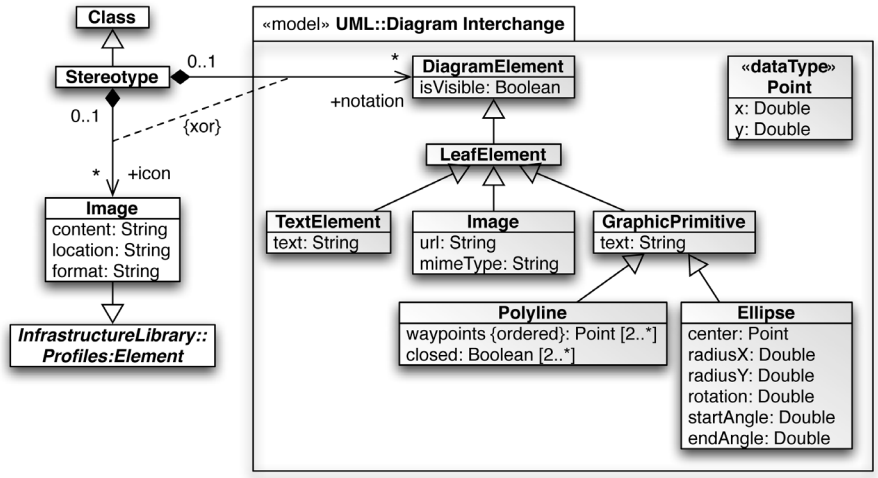


Figura 5. Extensión notacional de perfiles UML con el metamodelo DI

Nuestra propuesta consiste en incorporar a este metamodelo de notación elementos de modelado visual más potentes. Para ello, aquí proponemos a modo ilustrativo el uso del propio metamodelo de notación de UML, *i.e.*, el estándar para el intercambio de metadatos de diagramas o *Diagram Interchange* (DI) (esbozado en la Fig. 5). Este metamodelo ofrece elementos de modelado (metaclase `DiagramElement` y sus especializaciones) para representar notaciones complejas. Por ejemplo, los elementos de modelado pueden componerse a partir de diferentes nodos y aristas (`GraphNode` & `GraphEdge`, no mostrados en la figura por simplicidad).

Con este metamodelo de notación, los elementos de modelado de diagramas pueden representar textos (`TextElement`), imágenes (`Image`) o distintos tipos de primitivas gráficas (`GraphicPrimitive`). Nótese cómo, con este metamodelo, es todavía posible asociar sólo una imagen a una metaclase, lo que lo hace compatible con el modo actual de asociar iconos en los perfiles UML. Por último, queremos hacer notar que la relación {`xor`} de la Fig. 5 representa el carácter optativo (de manera exclusiva) de ambos metamodelos de extensión notacional (iconos o diagramas DI) con vistas a su implantación en el seno de herramientas de modelado.

Aplicación de perfiles UML mejorados

Esta propuesta de extensión notacional puede aplicarse a nuestro caso de estudio (§2) para solventar los problemas notacionales del perfil `ER--` (Fig. 3) sin recurrir a la elección de metaclases subóptimas para realizar la extensión. Para ello, en la Fig. 6 presentamos el perfil `ER++`. Este perfil se estructura en dos los paquetes: `ER++::Semántica` y `ER++::Notación`, para resaltar la existencia de tal separación de responsabilidades. Mientras que el subpaquete `Semántica` se define mediante la notación visual típica de los perfiles UML (similar a los diagramas de clases), el subpaquete `Notación` se define mediante diagramas de objetos que representan modelos DI. De esta forma, sin salirse del marco de UML, es posible definir notación y semántica por separado, cada una en base a las metaclases más apropiadas. Este 'mantenerse dentro del marco de UML' es la principal diferencia con respecto a la aproximación seguida por herramientas tan conocidas como *UML2 Tools*[4] o *MagicDraw*, donde, como ya hemos comentado, se utilizan paneles y diálogos complementarios para completar la especificación visual, limitando de este modo la compatibilidad de los perfiles definidos.

Como puede observarse en la Fig. 6, gracias a la separación de semántica y notación, el subpaquete `Semántica` correlaciona ahora los elementos de modelado del ER con metaclases UML más naturales que las elegidas en el perfil `ER--` (Fig. 3): entidades con clases, atributos con propiedades y relaciones con asociaciones. Como consecuencia, las restricciones OCL que acompañaban al perfil `ER--` quedan drásticamente simplificadas. Por otro lado, el subpaquete `Notación` permite enlazar a cada estereotipo el elemento de modelado, de complejidad arbitraria, correspondiente del metamodelo DI para definir dicha notación. Volviendo a nuestro caso de estudio, hemos relacionado `Entidad` con una `Polyline` que define un rectángulo, y `Relación` con una

[4] `www.eclipse.org/modeling/mdt/?project=uml2tools`

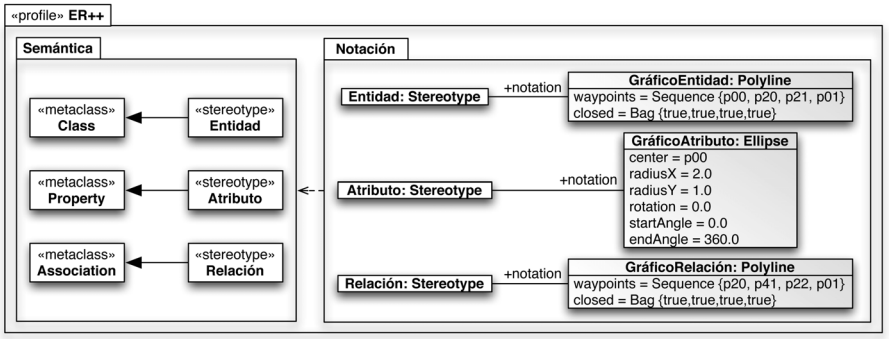


Figura 6. Perfil UML mejorado para diagramas ER

`Polyline` que define el rombo característico del ER. Nótese además que estas primitivas gráficas (Fig. 5) disponen de una propiedad `text` que permite representar la etiqueta de los nombres dentro del propio elemento gráfico.

Con esta especificación, cualquier herramienta de modelado configurada para manejar la extensión notacional aquí propuesta podría consultar la `notación` (Fig. 5) de los estereotipos definida en términos de modelos DI, en lugar de utilizar la notación convencional proporcionada por UML. En nuestro caso, esta extensión notacional nos permite crear perfiles para diagramas ER que reproducen *exactamente* la notación original (Fig. 2), que era nuestro principal objetivo. De manera similar, notaciones completamente nuevas como la de i^* pueden ser también reproducidas en un perfil UML mediante los elementos de diagramado proporcionados por el metamodelo DI, haciendo innecesarias las desviaciones semánticas (Fig. 4) y notacionales (Fig. 3) en las que a menudo se incurre.

4. Discusión

La separación de sintaxis/semántica y notación, lejos de ser nueva, es un principio básico en la ingeniería del software, y subyace en marcos de modelado como GMF [5], donde al igual que nosotros estudiamos para el perfilado de UML [6], la semántica y notación se modelan con metamodelos distintos que se enlazan mediante otros metamodelos diseñados para tal fin [6].

La propuesta presentada en este trabajo tiene implicaciones para UML que van mucho más allá de la mera alineación de notaciones de perfiles con notaciones de lenguajes de dominio originales. En relación a la capacidad expresiva de las notaciones diseñadas mediante perfiles UML, la inclusión de metamodelos específicos para la visualización de información como el aquí ilustrado, *Diagram Interchange*, evitaría algunos de los

[5] *Graphical Modelling Framework*, `www.eclipse.org/gmf`

[6] El análisis de perfiles contra metamodelos queda fuera del ámbito de este texto.

problemas más comunes con los que se enfrentan los diseñadores de perfiles y permitiría no sólo definir notaciones más flexibles para dichos perfiles, sino también la definición de técnicas de visualización alternativas para diagramas UML. Esto permitiría que un diseñador acostumbrado, por ejemplo, a la notación de diagrama de clases de Booch [3] pudiese seguir utilizando esa notación en herramientas de modelado UML, sin perder la interoperabilidad que proporciona el estándar.

Además, el proceso de extensión de perfiles a partir de dos metamodelos que cumplen propósitos distintos enfatiza las distintas finalidades de los perfiles, y da lugar a la siguiente clasificación:

perfiles semánticos aquellos cuyos estereotipos se definen sin notación extendida sobre la de UML

perfiles notacionales orientados a proveer únicamente cambios de notación, sin alterar la semántica de las metaclases base

perfiles híbridos los que especifican cambios semánticos y de notación

Es importante hacer notar que el metamodelo *Diagram Interchange* que estudiamos aquí está orientado a la visualización de diagramas. Por tanto, serían necesarios otros metamodelos de notación más expresivos si quisiéramos permitir otras técnicas de visualización como *e.g.* presentaciones basadas en matrices [11] que podrían ser necesarias por motivos de escalabilidad visual. En esta línea, la gestión de múltiples notaciones para una misma semántica podría requerir de elementos más expresivos para modelar su enlace [1]. Además, el uso de un metamodelo de notación extendido incrementa la curva de aprendizaje del diseñador de los perfiles. Por último, no debemos olvidar que muchas veces los perfiles UML se definen en el seno de una herramienta de modelado. Por tanto, para su definición exitosa, estas herramientas deberían guiar en la selección de metaclases y definición de notaciones asociadas a los perfiles de manera más efectiva. Esta dependencia de notaciones y entornos de modelado ha sido ampliamente reconocida en el campo de la interacción persona-ordenador, donde las *dimensiones cognitivas de las notaciones* [12] asumen que uno de los factores de éxito de dichas notaciones es precisamente el medio en el que se definen y manipulan, esto es, en nuestro caso, las herramientas de modelado UML.

Una línea de trabajo a desarrollar es la implementación de un motor de definición y manipulación de perfiles que soporte el modelado con *Diagram Interchange* de notaciones. Como paso posterior, se debe evaluar el grado de eficiencia, efectividad y satisfacción que conlleva en la práctica esta propuesta tanto para el diseño de perfiles UML como para su aplicación.

Referencias

1. Marcus Alanen, Torbjörn Lundkvist, and Ivan Porres. A Mapping Language from Models to DI Diagrams. In *MoDELS*, pages 454–468, 2006.
2. Stefan Berner, Martin Glinz, and Stefan Joos. A Classification of Stereotypes for Object-Oriented Modeling Languages. In *UML*, pages 249–264, 1999.
3. G. Booch. *Object-oriented analysis and design*. Addison-Wesley Reading, 1996.
4. P.P.S. Chen. The entity-relationship model–toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

5. T. Clark and A. Evans. Foundations of the unified modeling language. In *Proc. of the 2nd Northern Formal Methods Workshop. Springer*, 1997.
6. M.D. Del Fabro, J. Bezivin, F. Jouault, E. Breton, and G. Gueltas. AMW: a generic model weaver. In *Proc. of the 1ères Journées sur l'Ingénierie Dirigée par les Modèles*, 2005.
7. Brian Dobing and Jeffrey Parsons. How UML is used. *Commun. ACM*, 49(5):109–113, 2006.
8. Holger Eichelberger. Nice Class Diagrams Admit Good Design? In *SOFTVIS*, pages 159–167, 2003.
9. Holger Eichelberger and Klaus Schmid. Guidelines on the Aesthetic Quality of UML Class Diagrams. *Information and Software Technology*, In Press, 2009.
10. J.M. Favre. Towards a Basic Theory to Model Model Driven Engineering. In *3rd Workshop in Software Model Engineering, WiSME*, 2004.
11. Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. In *INFOVIS*, pages 17–24, 2004.
12. Thomas R. G. Green and Marian Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *J. Vis. Lang. Comput.*, 7(2):131–174, 1996.
13. Sergio Luján-Mora, Juan Trujillo, and Il-Yeol Song. A UML profile for multidimensional modeling in data warehouses. *Data Knowl. Eng.*, 59(3):725–769, 2006.
14. R.C. Martin. The Liskov substitution principle. *C++ Report*, 8(3):14–23, 1996.
15. Jose-Norberto Mazón, Jesús Pardillo, and Juan Trujillo. A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In *ER Workshops*, pages 255–264, 2007.
16. N. Moreno, P. Fraternali, and A.Vallecillo. A UML 2.0 Profile for WebML Modeling. *2nd Model-Driven Web Engineering Workshop, MDWE)*, 2006.
17. Object Management Group. Object Constraint Language (OCL), version 2.0. `http://www.omg.org/technology/documents/formal/ocl.htm`, October 2003.
18. OMG. Unified Modeling Language (UML) Superstructure, version 2.2. `http://www.omg.org/technology/documents/formal/uml.htm`, abril 2009.
19. Keng Siau and Qing Cao. How Complex Is the Unified Modeling Language? In *Advanced Topics in Database Research, Vol. 1*, pages 294–306. 2002.
20. W.M. Turski and T.S.E. Maibaum. *Specification of computer programs*. 1987.
21. R. Wirfs-Brock, B. Wilkerson, and L. Wiener. Responsibility-driven design: Adding to your conceptual toolkit. *ROAD*, 1(2):27–34, 1994.
22. Jose Zubcoff, Jesús Pardillo, and Juan Trujillo. A UML profile for the conceptual modelling of data-mining with time-series in data warehouses. *Information and Software Technology*, 51(6):977 – 992, 2009.

Decidable Reasoning in UML Schemas with Constraints

Anna Queralt and Ernest Teniente

Universitat Politècnica de Catalunya
Dept. de Llenguatges i Sistemes Informàtics
c/ Jordi Girona 1-3, 08034 Barcelona (Catalonia, Spain)
{aqueralt, teniente}@lsi.upc.edu

To ensure the final quality of an information system, it is essential that the conceptual schema that represents the knowledge about its domain is semantically correct.

The Unified Modeling Language (UML) has become a de facto standard in conceptual modeling of information systems. In UML, a conceptual schema is represented by means of a class diagram, with its graphical constraints, together with a set of user-defined constraints, which are usually specified in OCL. Due to the high expressiveness of the combination of both languages, checking the correctness of a UML conceptual schema manually becomes a very difficult task, specially when the set of textual constraints is large. For this reason, it is desirable to provide the designer with some automatic support to reason on a conceptual schema.

There are several reasoning tasks that can be performed to determine the correctness of a schema, such as satisfiability of the schema, liveliness of a class or association or reachability of certain states of the information base. Since the problem of reasoning with integrity constraints in its full generality is undecidable, two different approaches can be followed, either based on decision procedures for certain restricted kinds of constraints, or based on semidecision procedures for highly expressive constraints.

In this work we take a mixture of both directions. In particular, our approach consists in translating a UML schema, with OCL constraints satisfying mild syntactical restrictions, into a logic representation to reason on it. Then we provide a set of conditions to check that the schema does not have any infinite instantiation. In this case, any theorem prover or reasoning method can be used knowing that any reasoning task performed on the schema will terminate.

Additionally, once we have determined that all the instantiations of a schema are finite, and since the logic representation obtained from our UML and OCL schemas follows a specific syntactical structure, we provide a reasoning procedure that always terminates and works more efficiently than in the general case.

Abstract of the paper included in the Proceedings of the 20th international conference on Advanced Information Systems Engineering (CAiSE), pp. 281 - 295, 2008

Actas JISBD 2009, pp.354, ISBN: 978-84-692-4211-7

Rigorous Software Development Using McErlang*

Clara Benac Earle and Lars-Åke Fredlund

Grupo Babel, Facultad de Informática, Universidad Politécnica de Madrid
{lfredlund,cbenac}@fi.upm.es

Abstract. We present McErlang, a model checker for verifying distributed programs written in the Erlang programming language against LTL formulae.

1 Introduction

With the emergence of multicore technology, virtually all software will become concurrent. However, concurrent programming is difficult: parallel algorithms are often poorly understood, resulting in implementations with subtle bugs and high development costs. To combat this complexity, the software industry is starting to adopt rigorous development methods (hitherto mostly used in safety-critical systems and hardware design) also known as formal methods. In particular we have developed rigorous software engineering techniques, based on systematic model checking and testing, to improve reliability in distributed programs. Model checking can be seen as a complementary technique to testing for verifying concurrent systems. Model checking provides the possibility to, in theory, fully verify a system by exploring all its possible executions.

Erlang[1] is a functional programming language developed at Ericsson for the implementation of concurrent, distributed, fault-tolerant systems. In 1998 Erlang was released as Open Source. Today Erlang is used by Ericsson and many other companies (T-Mobile (UK), Swedish Telecom and many smaller start-up companies such as e.g. LambdaStream in A Coruña) to develop industrial applications (for example, a high-speed ATM switch developed at Ericsson, parts of Facebook chat, Apache CouchDB – a distributed, fault-tolerant and schema-free document-oriented database accessible via a RESTful HTTP/JSON API, users at Amazon, Yahoo!, etc.) with challenging requirements on software reliability and overall system availability.

To improve the reliability of Erlang programs we have developed McErlang, a model checker for Erlang. In contrast to most other Erlang verification attempts, we provide support for virtually the full, rather complex, programming language. The model checker has full Erlang data type support, support for general process communication, node semantics (inter-process communication behave subtly different from intra-process communication), fault detection and fault tolerance, and crucially can verify programs written using the high-level OTP Erlang component library (used by most Erlang programs).

* This work has been partially supported by the following projects: ProTest (FP7-ICT-2007-1 215868), DESAFIOS (TIN2006-15660-C02-02), and PROMESAS (S-0505/TIC/0407).

[1] www.erlang.org

For non-Erlang programmers the approach has merits too. Erlang is a good *general specification* language, due to its root in functional programming (with higher-order functions for writing concise specifications, with software components for lifting the specification abstraction level, can treat program as data – yielding a convenient meta level, ...). Moreover Erlang is a good platform for *specifying distributed algorithms* since language features matches common assumptions in distributed algorithms: *isolated* processes that communicate using message-passing, *fault-detection* is achieved using unreliable failure detectors, process *fairness* built into the language, and *locality* is important: communication guarantees are stronger for intra–node communication than inter–node communication.

2 Erlang

Erlang is a functional programming language developed at Ericsson for the implementation of concurrent, distributed and fault-tolerant systems. In the last couple of years the language has gained a lot of attention because of its concurrency oriented programming paradigm which matches well the multi-core processor hardware architecture that is becoming increasingly common.

Erlang provides a functional core language, which is enriched with the concept of *processes*. Processes are isolated from each other, communicating only via asynchronous message passing. Erlang processes have a unique name, a process identifier, and messages sent to the process are stored in the *process mailbox*. On top of the process concept the distribution layer of Erlang is implemented in terms of *nodes*. A node (e.g. a host runtime system) provides a distribution mechanism where processes are mapped onto different nodes. Both intra-node and inter-node communication are implemented, and the distribution aspect is mostly transparent; meaning that a process communicates in the same way with a local and a distributed process. A typical Erlang application is organized in many, relatively small, source components. At runtime the components execute as a dynamically varying number of processes running parallel (Erlang processes are very lightweight, and it is not uncommon having systems with several hundred thousand simultaneous processes.)

Handling a large number of processes easily turns into an unmanageable task, and therefore Erlang programmers mostly work with higher-level language components. The OTP component library offers industrially proven design patterns such as: a generic server component (for client-server communication), a finite state machine component, generic TCP/IP communication, and a supervisor component for structuring fault-tolerant systems. The Erlang programming system is nowadays often referred to as Erlang/OTP, to stress the benefit to distributed application programming that the OTP library provides.

3 The McErlang model checker

McErlang is a model checker, which has been used to verify a number of distributed Erlang applications. The model checker has recently been released as open source;

further details including information about how to download the tool is available at http://babel.ls.fi.upm.es/trac/McErlang/.

Figure 1 illustrates the differences between a normal Erlang workflow (left) and one using the McErlang model checker (right). The model is the Erlang program to be analyzed, which undergoes a source-to-source translation to prepare the program for running under the model checker. Then the normal Erlang compiler translates the program to either Beam byte code (an Erlang byte code language) or directly to native machine code. Finally the program is run under the McErlang run time system, under the control of a verification algorithm, by the normal Erlang bytecode interpreter. The pure computation part of the code, i.e, code with no side effects, including garbage collection, is executed by the normal Erlang run time system. However, the side effect part is executed under the McErlang run time system which is a complete rewrite in Erlang of the basic process creating, scheduling, communication and fault-handling machinery of Erlang (comprising a significant portion of the code of the model checker).

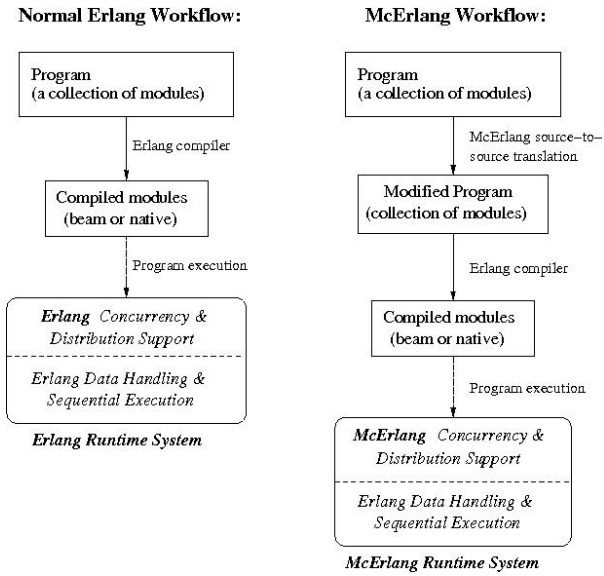


Fig. 1. Usage of McErlang

Naturally the new run time system offers easy check pointing (capturing the state of all nodes and processes, of the message queues of all processes, and all messages in transit between processes), of the whole program state as a feature (impossible to achieve in the normal Erlang run time system due to the physical distribution of processes).

The input to the model checker is an Erlang program, together with a special callback module (called a *monitor*) also written in Erlang, which specifies the behavioral

property to be checked (implementing either a safety automaton or a Büchi automaton). The output can be either a positive answer saying that the property holds, or a negative one together with a counterexample (typically an execution trace leading to a faulty program state).

The model checker implements full linear-temporal logic (LTL) checking; there is a automatic translator from LTL to their internal representation as Büchi automatons (*monitors* coded in Erlang). Properties are checked using a standard on–the–fly depth–first model checking algorithms, where the program under verification and the correctness automaton are run in lock–step. A *monitor* checks whether the correctness property holds for the combination of the new program state and the monitor state. If successful, the monitor returns an updated monitor state (for safety checking). Correctness properties can be implemented, therefore, as finite state machines where depending on the monitor state, transitions leading to new states are accepted or not. Such correctness properties have full access to the internal state of the system under verification (including message queues, state of processes, etc.).

McErlang has been successfully applied in a number of case studies, for example: a resource manager, a Video–on–demand server, leader election protocols, the control software for an elevator system, and multi-agent systems implementing soccer teams playing simulated robotic soccer.

As mentioned above Erlang has gained a lot of attention lately. Most of this is because of the inevitable change to *multi-core* systems. The concurrency oriented nature of Erlang together with the (almost) transparent distribution makes it a very good language for writing distributed/concurrent applications. Starting with release R11 (released in 2007) the Erlang runtime system has built-in support for SMP (Symmetric Multi Processing). The SMP support is fully transparent, and Erlang applications can therefore run on several cores without any changes. Even though the addition of SMP support to Erlang has proven straightforward compared to supporting SMP in languages such as e.g. Java, a number of bugs in both system and application software due to unwarranted assumptions regarding sequential execution have been uncovered. A tool like McErlang is ideally placed to discover such software race conditions, as it investigates all possible process schedulings.

4 Demonstration

The demonstration of McErlang will summarize the most important features of the Erlang programming language, and the use of the McErlang model checking tool to improve software and system reliability. Offline we will demonstrate concretely how to apply the tool to a number of complex concurrent and distributed systems: the control software for a set of elevators, a process-based registry application, and a messenger service, all implemented in Erlang.

UMLtoSBVR: Una herramienta para la validación de modelos UML mediante SBVR

Raquel Pau[1], Jordi Cabot[2] y Ruth Raventós[1]

[1] Universitat Politècnica de Catalunya (UPC)
raquelpau@gmail.com, raventos@lsi.upc.edu
[2] Estudis d'Informàtica, Multimedia i Telecomunicació, Universitat Oberta de Catalunya
jcabot@uoc.edu

Resumen. Presentamos UMLtoSBVR, una herramienta para parafrasear modelos UML, es decir, para describirlos en lenguaje natural. De esta forma, los modelos pueden ser validados por los expertos en el dominio que se está modelando, aunque éstos carezcan del perfil técnico necesario para entender directamente el modelo UML. Como paso intermedio, esta herramienta realiza una transformación de UML a SBVR (*Semantics of Business Vocabulary and Business Rules)*, un estándar que facilita la conversión de sus modelos a lenguaje natural.

Keywords: UML, SBVR, Validación de modelos, Transformación de modelos

1 Introducción

En un contexto de desarrollo de software dirigido por modelos (DSDM), la corrección de los modelos tiene un impacto directo en la calidad del código final (que se genera semi-automáticamente a partir de los primeros). Aunque hay diversos métodos para la *verificación* de modelos, se ha avanzado menos en el campo de la *validación* de los mismos. Validar implica determinar si el modelo representa correctamente la realidad de un dominio tal y cómo la perciben los expertos en dicho dominio. Sin embargo, éstos no tienen normalmente un perfil técnico con lo que les es muy difícil validar directamente los modelos UML definidos por diseñadores de software.

Una de las posibles soluciones a este problema, sugerida por *Sommerville* y *Sawyer* [1], consiste en *parafrasear* (es decir, describir utilizando lenguaje natural) modelos gráficos. De esta forma, un experto en el dominio puede validar el modelo sin tener que esperar a ver la implementación del sistema (mucho más costoso en tiempo y dinero). UMLtoSBVR[2] implementa esta solución vía una transformación intermedia del modelo UML a un modelo SBVR (*Semantics of Business Vocabulary and Business Rules*[3]), el cual es posteriormente convertido a lenguajes textuales como *Structured English* y *RuleSpeak* (simplificaciones del lenguaje inglés escrito). La herramienta implementa los resultados teóricos presentados en [4].

El propio documento de especificación del metamodelo SBVR[3], incluye algunas normas orientativas de cómo describir lo elementos de un modelo SBVR utilizando expresiones textuales siguiendo los patrones del *Structured English*. Sin embargo,

debido a su reciente aparición, aun no hay un visor estable para la versión actual de SBVR que implemente todas estas normas de representación. Consecuentemente, como parte del desarrollo de la herramienta UMLtoSBVR se ha implementado la transformación de modelos SBVR a expresiones de *Structured English*. Esta transformación puede ser de utilidad a cualquier otra herramientas que trabaje con modelos SBVR.

El uso de SBVR como representación intermedia permite enriquecer y mejorar los modelos UML (con por ejemplo, incluyendo definiciones y sinónimos que mejoren su comprensibilidad) desde un editor de SBVR así como independizar la transformación de la gramática usada para generar el lenguaje natural. Cambios en la gramática (por ej. pasando de usar el formato *Structured English* a *RuleSpeak*) no afectan a la transformación UML->SBVR.

2 Descripción de la herramienta

La herramienta recibe como input un diagrama de clases UML y genera como salida una página HTML que muestra el texto que describe el modelo en lenguaje natural siguiendo el formato *Structured English*. UMLtoSBVR está implementada dentro del entorno de *Eclipse Modeling Framework* (EMF[5]) como una doble transformación (modelo a modelo de UML a SBVR y modelo a texto de SBVR a Structured English) tal y como muestra la siguiente figura:

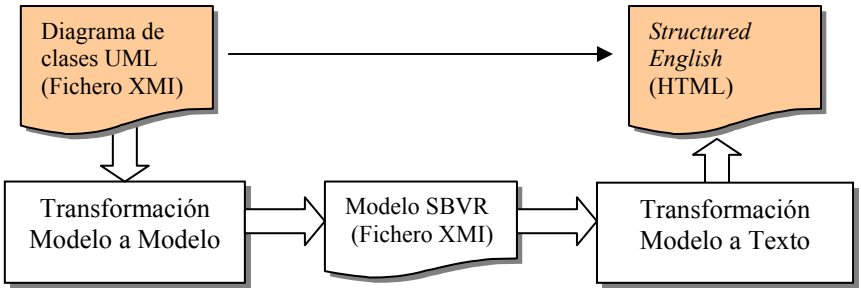


Fig. 1. Arquitectura de UMLtoSBVR

Transformación Modelo a Modelo. Ésta es una transformación intermedia de UML a SBVR que recibe como input un diagrama de clases modelado con el plugin UML2 (de EMF) y produce como salida un modelo SBVR equivalente. La transformación está definida usando el *Atlas Transformation Language*(ATL)[6] tal y como nuestra la figura siguiente(Fig. 2). El metamodelo de SBVR usado como destino de la transformación está actualmente en desarrollo como parte del componente MDT/SBVR de Eclipse[7]. En nuestra herramienta se ha completado ligeramente la versión actual del metamodelo para poder ofrecer una transformación completa del modelo UML inicial. No obstante, debido a las limitaciones de la versión actual del metamodelo de SBVR (que no incluye la parte del estándar

referente a la definición de las reglas de negocio) las restricciones del modelo UML se omiten durante la transformación.

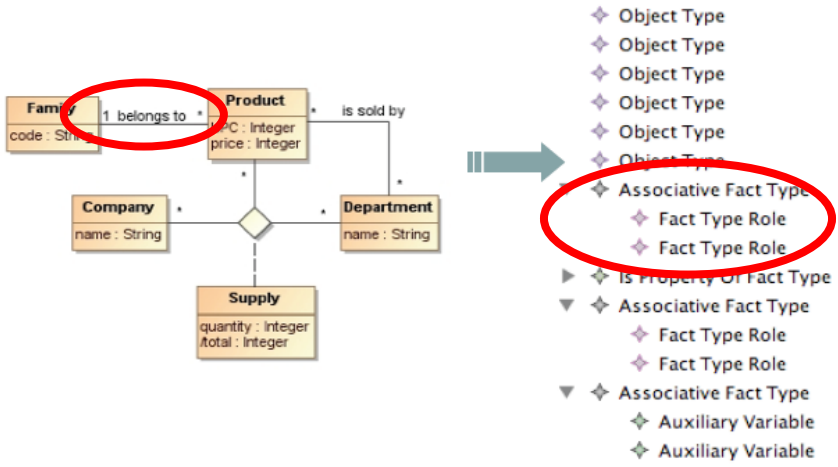


Fig. 2. Resultado(parcial) de un modelo SBVR generado a partir de un diagrama de clases.

Transformación Modelo a Texto. A partir del modelo SBVR generado en la transformación anterior, aplicamos una transformación modelo a texto para generar la descripción textual en *Structured English* usando *MOFScript* [8]. Esta transformación genera en código HTML el glosario de conceptos (clases y asociaciones) existentes en el modelo SBVR juntamente con sus definiciones, sinónimos, etc. (si estos se han añadidos al modelo SBVR), como se muestra en el ejemplo de la Fig. 3.

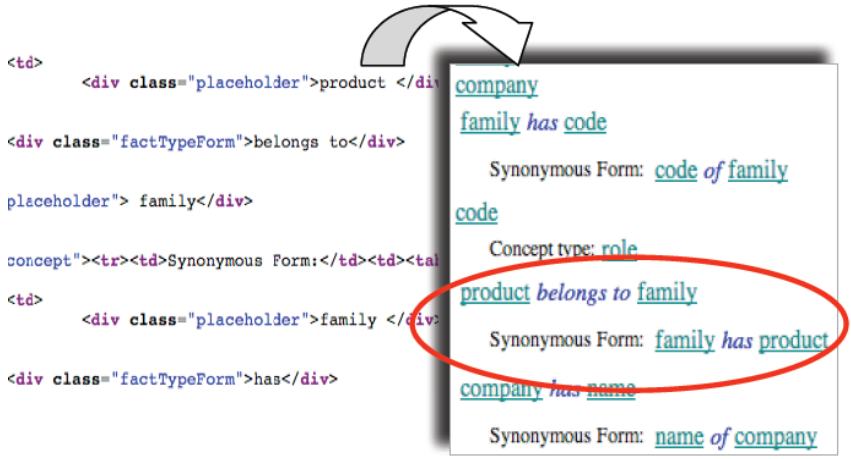


Fig. 3. Resultado(parcial) de la transformación SBVR a Structured English

3 Trabajo futuro

Se está trabajando para ampliar la herramienta en varios aspectos. En primer lugar, se está diseñando un conjunto de transformaciones ATL para transformar las restricciones de integridad del modelo inicial (tanto las expresadas en UML, como por ej. *xor*, *subset*, etc., como las definidas textualmente utilizando el lenguaje OCL). En segundo lugar, se está completando el metamodelo de SBVR actualmente en desarrollo como parte del proyecto MDT/SBVR para que incluya todas las clases definidas en la especificación de SBVR. Esto es necesario parar poder expresar cualquier tipo de modelo UML/OCL en SBVR. Por lo que respecta a la transformación modelo a texto, se trabajará para, primero, generar descripciones en *Structured English* a partir del metamodelo SBVR completo y para, segundo, añadir soporte para otros lenguajes (como el español) de forma, que el output de la herramienta pueda ser multi-lingüe. Por último, se pretende desarrollar un *plug-in* de Eclipse que automatice la ejecución de toda la cadena de transformaciones de cara a facilitar el uso de la herramienta.

Referencias

1. I. Sommerville, P. Sawyer, Requirements engineering: a good practice guide. John Wiley & Sons, Inc. New York, NY, USA. 1997
2. J. Cabot, R. Pau, R. Raventós, UML-to-SBVR and SBVR-to-HTML transformations. http://jordicabot.com/research/SBVR/index.html
3. OMG: Semantics of Business Vocabulary and Rules (SBVR) Specification, v. 1.0 formal/08-01-02), 2008
4. J. Cabot, R. Pau, R. Raventós, From UML/OCL to SBVR Specifications: a Challenging Transformation. Information Systems Elsevier Journal.
5. Eclipse Modeling Framework. http://www.eclipse.org/emf
6. ATL Project http://www.eclipse.org/m2m/atl/
7. MDT/SBVR Proposal http://wiki.eclipse.org/MDT-SBVR-Proposal
8. MOFScript Project http://www.eclipse.org/gmt/mofscript/

Una Aplicación basada en Eclipse para la Personalización de Aplicaciones Web Dirigida por Modelos

Irene Garrigós, Octavio Glorio, Paul Hernández, and Alejandro Mate

Lucentia Research Group
Departamento de Lenguajes y Sistemas Informáticos – DLSI
Universidad de Alicante, España
{igarrigos,oglorio,phernandez,amate}@dlsi.ua.es

1 Introducción

Las aplicaciones Web personalizadas tienen como principal objetivo proveer al usuario una vista específica que satisfaga sus necesidades y metas en el sitio Web. De este modo, aumenta la satisfacción del usuario, ya que el contenido, navegación y presentación se ajusta a sus requisitos y preferencias encontrando fácilmente lo que busca.

Las metodologías de diseño Web proveen una aproximación sistemática para el desarrollo de aplicaciones Web, y muchas de ellas soportan de algún modo el modelado de la personalización. Sin embargo, ninguna de estas metodologías proveen una herramienta que permita el modelado, generación automática y despliegue de aplicaciones Web personalizadas de una forma completa. Del mismo modo, tampoco existen herramientas para la modificación por parte del diseñador de las estrategias de personalización en tiempo de ejecución, evitando regenerar la aplicación Web desde cero. Como ejemplo, podemos mencionar la herramienta asociada al método de diseño Hera (HPG) [1], en el cual se especifican condiciones de visibilidad sobre los elementos de los modelos de diseño de Hera. Esta herramienta no realiza la instalación de la aplicación Web generada, y tampoco permite la modificación en tiempo de ejecución de estas condiciones de visibilidad, ya que están embebidas en el código de las páginas Web. Otro ejemplo es WebRatio [2] desarrollada para soportar la metodología WebML, sólo considera la especificación de personalización estática (en tiempo de diseño) con respecto a las características del usuario o el dispositivo, por lo que la generación de este tipo de aplicaciones es similar a la generación de una aplicación Web tradicional (sin personalización).

La herramienta que se presenta en este artículo se ha creado para dar soporte al método de diseño A-OOH (*Adaptive Object Oriented Hypermedia*) [3]. Esta aproximación es la extensión del método de modelado OO-H [4], con soporte de personalización. El principal objetivo de la herramienta es la generación automática de aplicaciones Web personalizadas a partir de modelos conceptuales (definidos con A-OOH) y su posterior despliegue. Otro de los objetivos principales es el mantenimiento de la personalización en tiempo de ejecución.

Los fundamentos de esta herramienta fueron publicados en [5]. En la actualidad, la herramienta ha sido mejorada añadiéndose editores gráficos para los modelos en

Actas JISBD 2009, pp.363-366, ISBN: 978-84-692-4211-7

Eclipse, así como permitiendo una instalación automática de la aplicación Web personalizada y los módulos necesarios una vez generada. Además, se ha incluido en la herramienta la posibilidad de definir un modelo de requisitos para la aplicación Web presentado en [6], cuyo objetivo final es la generación automática de un primer esqueleto de los modelos conceptuales que el diseñador sólo deberá refinar.

2 Desarrollo de Aplicaciones Web Personalizadas dirigido por Modelos

Típicamente, para definir una aplicación Web debemos definir tres modelos de diseño principales: un *modelo de dominio*, en el cual la estructura de los datos de dominio se define, un *modelo de navegación*, en el cual la estructura y el comportamiento de la vista de navegación sobre los datos de dominio se define, y finalmente un *modelo de presentación*, en el cual se define la presentación de la aplicación. Para modelar la personalización en tiempo de diseño se necesitan dos modelos adicionales: un *modelo de personalización*, en el que se especifican las estrategias de personalización, y un *modelo de usuario* en el que se especifica la estructura de la información necesaria para la personalización.

En [6] se ha añadido al conjunto de modelos de A-OOH, un *modelo de requisitos* basado en i* [7] para la especificación de los requisitos de usuario. Con el fin de que los requisitos sean consistentes con el diseño se pretende realizar transformaciones entre modelos con el objetivo de generar un esqueleto de los modelos conceptuales de la aplicación Web. Actualmente tenemos definido un conjunto de reglas de transformación para ciertos modelos conceptuales, pero al no estar completo este módulo se encuentra fuera de la herramienta, pudiéndose no obstante, definir el modelo de requisitos de la aplicación Web.

La herramienta presentada toma como *input* el conjunto de modelos de diseño de A-OOH, en formato XMI [8] y el motor de generación obtiene el siguiente *output*:

- Aplicación Web: Se genera un conjunto de páginas Web en ASP.NET.
- Base de datos de la aplicación: los modelos conceptuales son mapeados a una base de datos orientada a objetos.
- Módulos para gestionar la personalización: concretamente un motor Web encargado de gestionar los eventos, un módulo encargado de evaluar y ejecutar las reglas de personalización modificando los modelos conceptuales y generando la(s) página(s) o parte(s) necesaria(s) y un módulo para la gestión de la personalización en tiempo de ejecución.

2.1 Implementación

La herramienta propuesta se ha integrado en la plataforma de desarrollo ECLIPSE [1] tal como se muestra en la Fig. 1. ECLIPSE es un proyecto de código abierto concebido como una plataforma modular que puede extenderse mediante plugins para añadir nueva funcionalidad y características. De este modo, hemos diseñado una serie de plugins que

[1] URL: `http://www.eclipse.org`

nos permiten diseñar los modelos conceptuales y a partir de ellos generar una aplicación Web personalizada.

El modelo de personalización, consiste en un conjunto de reglas Evento-Condición-Acción, que se definen en el lenguaje PRML(*Personalization Rules Modeling Language*) [9] de un modo ortogonal al resto de los modelos. Estas reglas se sitúan en un fichero de texto, de una forma separada al resto de la aplicación de tal modo que son fácilmente modificables.

Por un lado, para dar soporte al modelado, el mecanismo de perfiles de UML(*Unified Modeling Language*) ha sido utilizado y se han desarrollado editores gráficos basados en estos perfiles. Estos editores se han construido sobre el marco nativo de modelado de Eclipse (EMF) y su extensión gráfica (GMF). Por otro lado, para dar soporte a la generación, el plugin de Eclipse utiliza servicios Web (*Java*) y el motor de generación que se ha desarrollado para la derivación de código (*C#*). Finalmente, para dar soporte a la instalación de la aplicación Web generada, la herramienta cuenta con un instalador que despliega la aplicación en un servidor. Una vez la aplicación está operativa, las reglas de personalización pueden ser actualizadas en tiempo de ejecución en un editor de texto integrado.

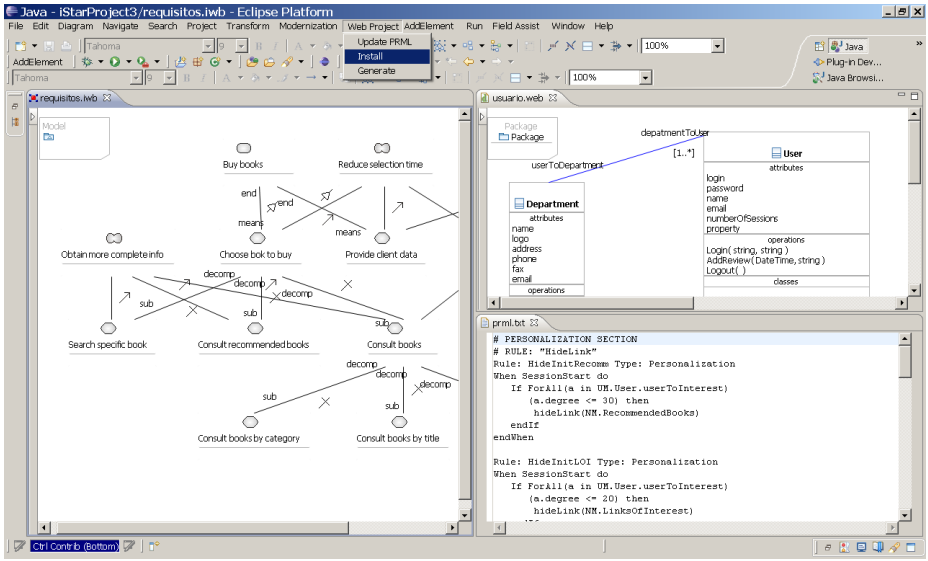


Fig. 1. Captura de la Herramienta

En la Fig 1 se puede ver una captura de la herramienta. En ella se observan diferentes modelos de diseño dentro de sus respectivos editores. También se puede ver un editor de las reglas de personalización. Finalmente, en la parte superior se encuentra el menú que controla la generación e instalación de la aplicación, así como la actualización de las reglas de personalización.

Agradecimientos Este trabajo ha sido parcialmente subvencionado por el proyecto ESPIA (TIN2007-67078) del Ministerio de Educación y Ciencia, y por el proyecto QUASIMODO (PAC08-0157-0668) del Ministerio de Educación y Ciencia de Castilla-La Mancha.

References

1. Frasincar, F., Houben, G.J., Barna, P.: Hera presentation generator. In: WWW (Special interest tracks and posters). (2005) 952–953
2. Acerbis, R., Bongio, A., Butti, S., Ceri, S., Ciapessoni, F., Conserva, C., Fraternali, P., Toffetti, G.: Webratio, an innovative technology for web application development. In: ICWE. (2004) 613–614
3. Garrigós, I.: A-OOH: Extending Web Application Design with Dynamic Personalization. PhD thesis, University of Alicante, Spain (2008)
4. Cachero, C., Gómez, J.: Advanced conceptual modeling of web applications: Embedding operation interfaces in navigation design. In: JISBD. (2002) 235–248
5. Garrigós, I., Cruz, C., Gómez, J.: A prototype tool for the automatic generation of adaptive websites. In: AEWSE. (2007)
6. Garrigós, I., Mazón, J.N., Trujillo, J.: A requeriment analysis approach for using i* in web engineering. In: ICWE, in Press. (2009)
7. Yu, E.: Modelling Strategic Relationships for Process Reenginering. PhD thesis, University of Toronto, Canada (1995)
8. XML Metadata Interchange: `http://www.omg.org/technology/documents/formal/xmi.htm`
9. Garrigós, I., Gómez, J.: Modeling user behaviour aware websites with prml. In: WISM. (2006)

Takuan: generación dinámica de invariantes en composiciones de servicios web con WS-BPEL

Manuel Palomo Duarte, Antonio García Domínguez, Alejandro Álvarez Ayllón y Inmaculada Medina Bulo

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Cádiz
C/ Chile nº 1, CP 11002, Cádiz (España)
{antonio.garciadominguez,manuel.palomo,
alejandro.alvarez,inmaculada.medina}@uca.es

Resumen La realización de pruebas de caja blanca de composiciones de servicios web con el estándar OASIS WS-BPEL 2.0 presenta un reto para las técnicas clásicas de prueba de software, debido a la inclusión de instrucciones específicas para manejar concurrencia, compensación de fallos, etc. Takuan es una herramienta que hemos desarrollado para la generación automática de invariantes de composiciones WS-BPEL a partir de trazas de ejecución reales. En este artículo presentamos sus posibilidades.

1. Introducción

Los servicios web y las arquitecturas orientadas a servicios parecen ser una de las claves para entender el futuro de la informática. Entre ellas destaca el estándar OASIS WS-BPEL 2.0 [1], que permite componer servicios más potentes basados en otros disponibles. Pero la aplicación de técnicas de prueba de caja blanca a composiciones WS-BPEL representa un reto, debido a la inclusión [2] de instrucciones específicas para el manejo de servicios web, como gestión de compensación, fallos, etc.

La generación automática de invariantes potenciales [3] ha demostrado ser una técnica eficaz para ayudar en la prueba y mejora de programas escritos en lenguajes imperativos tradicionales. Conviene aclarar en este punto que en este artículo se utilizan los términos «invariante» e «invariante potencial» en el mismo sentido en que se usa en la bibliografía de la materia [3], refiriéndose «invariante» a cualquier propiedad que es cierta en un determinado punto del programa (como un aserto, pre-condición, invariante de bucle, etc), e «invariante potencial» a cualquier propiedad que se mantiene en los casos de prueba ejecutados.

En este artículo presentamos Takuan [4,5,6], una herramienta que genera dinámicamente invariantes potenciales a partir de información recopilada en diversas ejecuciones de una composición WS-BPEL sobre un motor real. Para ello toma como entrada los ficheros de definición del proceso y un conjunto de casos de prueba, y da como salida un conjunto de invariantes que se cumplen en diversos puntos del programa para todos los casos de prueba.

Actas JISBD 2009, pp.367-370, ISBN: 978-84-692-4211-7

2. Invariantes y Composiciones WS-BPEL

2.1. Por qué generación dinámica de invariantes en WS-BPEL

Hasta la fecha se ha investigado poco sobre la aplicación de técnicas de prueba de caja blanca directamente sobre código de composiciones WS-BPEL ejecutado en un entorno real. Las principales propuestas [2] crean un modelo de simulación en un entorno especializado para pruebas.

Pero la simulación de un motor WS-BPEL es algo complejo, dado que hay una gran cantidad de características nada triviales que implementar (como el descubrimiento e invocación de servicios con determinadas propiedades en tiempo de ejecución). En caso de que alguna de estas características no se implementara correctamente, la composición no se estaría probando adecuadamente. Por ello, consideramos que el uso de modelos es una técnica propensa a errores, dado que no se basa en la ejecución del código WS-BPEL en un entorno real.

Por contra, Takuan genera dinámicamente invariantes potenciales a partir de las trazas de una serie de ejecuciones de la composición WS-BPEL en un motor real invocando servicios reales. Por lo tanto, se adapta mejor a la naturaleza del lenguaje y es una ayuda interesante para la prueba de programas WS-BPEL.

2.2. Uso de invariantes

Los invariantes generados a partir de un programa pueden usarse de diversas formas para mejorarlo. Por ejemplo, un invariante inesperado puede hacernos ver un fallo en el código que de otra forma podría haber pasado desapercibido. También, a la hora de modificar un programa, se pueden comprobar qué invariantes deben mantenerse y cuáles no entre dos versiones de él (por lo que cualquier diferencia indicaría que se ha introducido algún error en el nuevo código). Incluso se puede comparar la especificación del programa con los invariantes obtenidos para ver si ésta se cumple.

Además, la obtención dinámica de invariantes potenciales permite evaluar y mejorar un conjunto de casos de prueba si se dispone de la especificación de un programa. Eso es debido a que el uso de un conjunto de casos de prueba con deficiencias (de cobertura, por ejemplo) puede provocar la aparición de invariantes erróneos, pues el motor de inferencia de invariantes recibirá información parcial. Por lo tanto, al obtener invariantes falsos de un generador dinámico como Takuan, hay que comprobar si dichos invariantes son debidos a fallos en el programa o a deficiencias en el conjunto de casos de prueba usado para inferirlos. Para ello basta con incorporar casos de prueba específicos para ello en posteriores ejecuciones, mejorando de esta forma el conjunto de casos de prueba original, y observar si se siguen infiriendo o no dichos invariantes falsos.

3. Takuan

Takuan usa varios sistemas libres que han sido modificados e integrados [4]: BPELUnit como biblioteca de pruebas unitarias, ActiveBPEL como motor WS-BPEL y Daikon como generador de invariantes.

Takuan es software libre, y se puede descargar gratuitamente de su web oficial [7]. En dicha web proporcionamos tanto el código fuente como una imagen de máquina virtual para disponer fácilmente en cualquier equipo de un sistema completo con la última versión de Takuan configurada.

Para facilitar el uso de Takuan hemos desarrollado un plugin para NetBeans (uno de los entornos de desarrollo WS-BPEL más populares) que permite el uso desde un asistente con interfaz gráfica, como se observa en la figura 1.

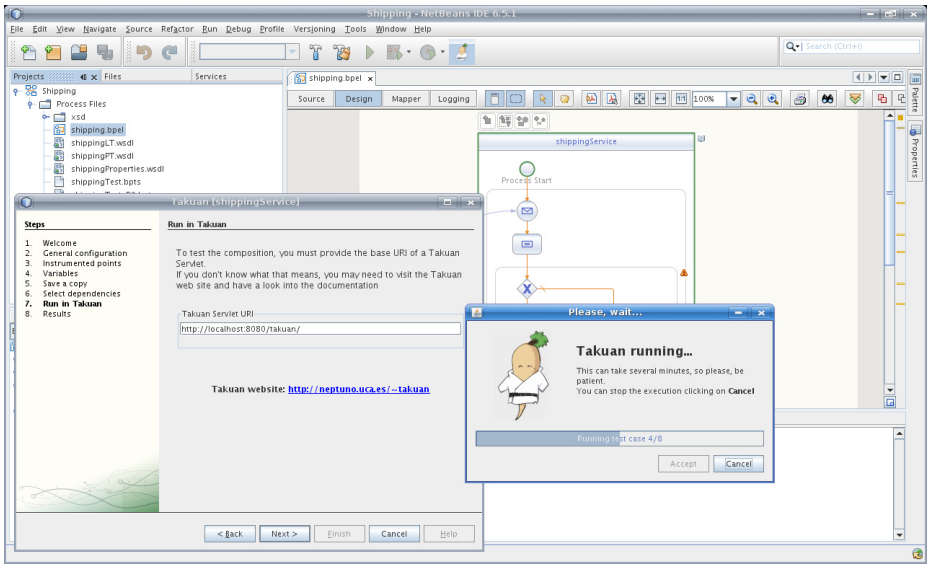


Figura 1. Plugin de Takuan para NetBeans.

Por defecto, Takuan toma toda secuencia WS-BPEL como punto del programa para comprobar invariantes, y en ella registra todas las variables. Sin embargo, también permite seleccionar manualmente aquellos puntos y variables sobre los que operar.

Además, permite sustituir las invocaciones a servicios externos por otras a servicios simulados. Esta opción sólo la recomendamos en caso de servicios que presenten limitaciones (por uso de recurso, costes, etc) o para probar el comportamiento de una composición en un escenario con determinadas respuestas de servicios. En este caso los servicios simulados se comportarán de acuerdo a la especificación proporcionada por el usuario para el caso de prueba (pudiendo devolver un mensaje predeterminado o dar un fallo concreto si así se ha indicado), siendo él responsable de proporcionar valores adecuados.

Takuan se ha aplicado con éxito a diversas composiciones, como el ejemplo del préstamo bancario incluido en el estándar WS-BPEL [4] o una composición de metabúsqueda de información [5]. Tras dichas pruebas se observaron ciertas limitaciones en sus prestaciones, por lo que se desarrollaron una serie de mejoras tanto de consumo de

recursos (disminuyendo el uso de CPU y memoria, lo que permite analizar composiciones más grandes) y de mejora de la salida, evitando invariantes redundantes [6].

4. Conclusiones y Trabajo Futuro

En este artículo hemos presentado Takuan, una herramienta que genera dinámicamente invariantes potenciales a partir de una composición WS-BPEL y un conjunto de casos de prueba. Con ella podemos superar las dificultades que la aplicación de técnicas tradicionales de prueba de caja blanca presenta para WS-BPEL. Esto es, en gran medida, gracias a que la generación de invariantes está basada en la información de registros obtenidos a partir de ejecuciones reales de la composición en un motor real.

Nuestro siguiente paso será comprobar Takuan con más composiciones y estudiar la relación entre la calidad de los invariantes inferidos por Takuan y la del conjunto de casos de prueba empleado para generarlos.

Agradecimientos

Este trabajo ha sido financiado por el Programa Nacional de I+D+I del Ministerio de Educación y Ciencia y fondos FEDER mediante el proyecto SOAQSim (TIN2007-67843-C06-04).

Referencias

1. OASIS: WS-BPEL 2.0 standard. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (abril 2007)
2. Bucchiarone, A., Melgratti, H., Severoni, F.: Testing service composition. En: Proceedings of the 8th Argentine Symposium on Software Engineering (ASSE'07). (2007)
3. Ernst, M.D., Cockrell, J., Griswold, W.G., Notkin, D.: Dynamically discovering likely program invariants to support program evolution. IEEE Transactions on Software Engineering **27**(2) (febrero 2001) 99–123
4. Palomo Duarte, M., García Domínguez, A., Medina Bulo, I.: Takuan: A dynamic invariant generation system for WS-BPEL compositions. En: ECOWS '08: Proceedings of the 2008 Sixth European Conference on Web Services, Washington, DC, USA, IEEE Computer Society (2008) 63–72
5. Palomo Duarte, M., García Domínguez, A., Medina Bulo, I.: Improving Takuan to analyze a meta-search engine WS-BPEL composition. En: SOSE '08: Proceedings of the 2008 IEEE International Symposium on Service-Oriented System Engineering, Washington, DC, USA, IEEE Computer Society (2008) 109–114
6. Palomo-Duarte, M., García-Domínguez, A., Medina-Bulo, I.: Enhancing WS-BPEL dynamic invariant generation using XML Schema and XPath Information. En: ICWE'09: Proceedings of the 9th International Conference on Web Engineering. Volume 5648 of Lecture Notes in Computer Science., Springer (2009) 469–472
7. Grupo SPI&FM: Web oficial de Takuan. http://neptuno.uca.es/˜takuan

Parte XI

Sesión 10. Calidad, Medición y Estimación de Productos y Procesos Software

Un análisis de la Calidad en Uso de los Componentes Software utilizando Redes Bayesianas

M.F. Bertoa[1], M.A. Moraga[2], M.C. Morcillo[3], C. Calero[2]

[1]Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga, Spain
[2]Dept. Information Technologies and Systems. Universidad de Castilla-La Mancha, Spain
[3]Dept. Estadística e Investigación Operativa. Universidad de Málaga, Spain
bertoa@lcc.uma.es, mariaangeles.moraga@uclm.es, aixela@uma.es, coral.calero@uclm.es

Resumen. Las propuestas para mejorar la calidad de los productos software se han centrando tradicionalmente en mejorar la Calidad Interna o Externa (independientemente de los posibles contextos de uso), basándose en la idea de que una buena Calidad Externa (según los términos de ISO/IEC 9126) garantiza un buena Calidad en Uso. Este artículo plantea otro camino, cambiando el foco de atención hacia la Calidad en Uso como elemento determinante en el diseño de productos software de alta calidad para usuarios concretos o para seleccionar el producto que mejor se ajuste a las necesidades de un cliente. Nuestra propuesta consiste en realizar un análisis de las relaciones entre la Calidad Externa y la Calidad en Uso para determinar las subcaracterísticas de Calidad Externa que son realmente relevantes para asegurar el nivel requerido de calidad de un producto en un contexto de uso específico. El objetivo es evitar costes superfluos o características irrelevantes para el usuario final que incrementan innecesariamente el coste y el esfuerzo de desarrollo de los productos. En este trabajo proponemos las Redes Bayesianas para modelar esas relaciones y ofrecer un método para definirlas de una forma cuantificable. Como ejemplo de uso del método propuesto nos hemos centrado en el domino de los componentes software para realizar un análisis de su Calidad en Uso.

Keywords: Calidad del software, Calidad en uso, Redes bayesianas, Componentes software.

1 Introducción

Valorar la calidad de productos software es, en general, una tarea difícil y compleja. Un enfoque para simplificar este proceso de evaluación consiste en descomponer la calidad del producto en varios factores o características. Una de las propuestas más extendidas para evaluar la Calidad del software es la definida por la norma ISO/IEC 9126 [5]. Este estándar internacional establece tres niveles (o vistas) en los cuales la calidad del producto software puede ser observada: Interna, Externa y en Uso (Fig. 1).

La Calidad Interna utiliza una aproximación de "caja blanca" del producto software, relacionada principalmente con sus propiedades estáticas y se evalúa durante las fases de diseño y desarrollo. La Calidad Externa usa una propuesta de "caja negra", valorando el producto software según sus atributos y características externas; es decir, aquellos que pueden ser percibidos por sus usuarios. Finalmente, la Calidad en Uso es

el punto de vista del usuario final cuando el producto software se utiliza en su entorno de trabajo, para llevar a cabo las tareas específicas que el usuario final necesita realizar. Entre estas *Calidades*, o puntos de vista de la Calidad, hay una fuerte relación, como se muestra en la Figura 1.

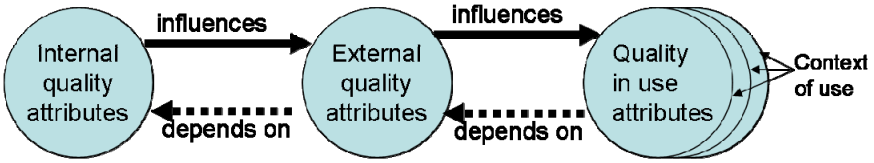


Fig. 1. Relaciones entre la vistas de la Calidad según ISO/IEC 9126 [5].

Hasta ahora, el énfasis de la comunidad de Calidad del Software ha estado puesto en la valoración de la Calidad Interna y Externa de los productos software (véase [1-3, 6, 8, 10, 11], principalmente porque estas dos vistas están definidas de forma más precisa en la literatura, se pueden identificar más fácilmente y, por tanto, son más fáciles de evaluar.

Esta relación entre la Calidad Externa y en Uso de los productos software parece estar basada en la presunción de que tener un producto con una alta calidad (externa) garantiza un producto con una alta Calidad en Uso. Sin embargo, esto no es necesariamente cierto en una mayoría de situaciones: todos conocemos que un producto con la mejor calidad (externa) no garantiza necesariamente que dicho producto satisfará las necesidades del usuario en su entorno de uso. Especialmente cuando la calidad global (como la percibe el usuario final) está compuesta de muchos factores en conflicto; por ejemplo, un Ferrari no es el mejor coche para ir al trabajo si eres un asistente social que trabaja en un barrio marginal de una gran ciudad.

La Calidad Externa de un producto software es importante, pero en este trabajo cambiamos el foco de atención hacia la Calidad en Uso, ya que la vemos como el factor clave a considerar cuando se diseña un producto para un mercado específico o cuando se selecciona el producto que mejor satisface las necesidades de un cliente.

Hay varias razones que nos han motivado a cambiar la forma de evaluar la calidad de un producto software. En primer lugar, no todas las características de Calidad Externa tienen la misma influencia sobre la Calidad en Uso. Esto, junto con la falsa creencia acerca de la dependencia directa entre la Calidad Externa y la Calidad en Uso, antes mencionada, fuerza muchas veces a sobre-especificar algunos aspectos de los productos, que no son críticos para los usuarios finales con la intención de asegurar un cierto nivel de Calidad en Uso. Esta decisión incrementa innecesariamente los costes y el esfuerzo de desarrollo, sin un efecto proporcional sobre las ventajas que percibe el usuario final.

En segundo lugar, los usuarios normalmente tienden a percibir con igual importancia todas las características de Calidad Externa de un producto software cuando son preguntados acerca de ellos en abstracto; es decir, sin ningún contexto de uso en mente. Sin embargo, cuando se fija un contexto de uso para el producto software, los usuarios son capaces de discriminar las características del producto que realmente les interesan en dicho contexto de aquellas que son deseables pero no imprescindibles desde su punto de vista. De esta forma, la calidad "real" del producto dependerá del contexto de uso; es decir, desde el punto de vista de un coche como una

"caja negra", el Ferrari es el coche que tiene mejor calidad, sin embargo, esto cambia cuando a ese coche lo ponemos en un contexto de uso determinado.

Este artículo se centra en las relaciones entre la Calidad Externa y la Calidad en Uso de un producto software. Así, se propone un análisis hacia atrás que ayude a seleccionar un grupo reducido de subcaracterísticas de Calidad Externa que sean *verdaderamente relevantes* y que aseguren el nivel de Calidad en Uso requerido, centrándose únicamente en ellas para evitar costes superfluos y aspectos irrelevantes que incrementan innecesariamente el precio final del producto.

Este trabajo analiza las diferentes posibilidades para modelar esas relaciones y proporciona una propuesta que se puede usar en diferentes contextos de uso. Para ello se usan Redes Bayesianas para representar las relaciones entre la Calidad Externa y la Calidad en Uso. Nuestro enfoque propone un modelo para los componentes software dejando para futuros trabajos extender y validar la propuesta para otros dominios diferentes.

Este artículo está organizado como sigue. Tras esta introducción, la sección 2 presenta el modelo de calidad de ISO/IEC 9126, que hemos utilizado como marco de trabajo de calidad en nuestro estudio. A continuación, en la sección 3,se da una breve introducción de las Redes Bayesianas. La sección 4 propone cómo construir una Red Bayesiana para modelar la Calidad en Uso y la sección 5 describe la construcción y utilización de una Red Bayesiana para evaluar la Calidad en Uso de componentes software. Finalmente, la sección 6 discute algunas conclusiones y propone líneas para futuras investigaciones.

2 El modelo de calidad de ISO/IEC 9126

ISO/IEC 9126 [5] propone uno de los modelos de calidad más ampliamente usado, descomponiendo la calidad (global) de un producto software en tres vistas (*Interna*, *Externa* y *En Uso*) y definiendo cada una en términos de características y subcaracterísticas. ISO/IEC 9126 define 6 características para calidad Interna y Externa: *Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad* y *Portabilidad*. Es de destacar que estas características son las mismas en ambos casos porque el estándar define un paralelismo entre ellas, asumiendo que existe una relación de dependencia una a una (p.e.: la Fiabilidad Interna influye en la Fiabilidad Externa y viceversa). Cada una de las características del modelo de calidad se refina en varias subcaracterísticas para poder realizar un análisis de valoración más preciso.

Respecto a la Calidad en Uso, ISO/IEC 9126 no define un paralelismo similar con la Calidad Externa. De hecho, se definen sólo cuatro características para la Calidad en Uso: *Efectividad, Productividad, Seguridad y Satisfacción*, las cuales no se descomponen en subcaracterísticas. ISO/IEC 9126 no incluye cómo representar la relación de influencia entre las seis características de Calidad Externa y las cuatro de Calidad en Uso, que es precisamente el objetivo de nuestro trabajo.

Finalmente, debemos señalar que ISO está trabajando actualmente en la preparación de una nueva familia de normas ISO 25000 que ha recibido el nombre de SQuaRe (Software QUAlity REquirements), que sustituirán entre otros al estándar 9126. Sin embargo, SQuaRe no está aún maduro. Realmente, no hay todavía consenso en el *Working Group* sobre las diferentes vistas de la calidad que quieren distinguir. Por esta causa, hemos decido basar nuestra propuesta en un estándar internacional

publicado como es ISO/IEC 9126 y esperar a que la propuesta de SQuaRE se estabilice. En cualquier caso, la propuesta que presentamos para construir una Red Bayesiana se podría adaptar y construir sin mayor dificultad para el nuevo Modelo de Calidad de SQuaRe, algo que entra en nuestros objetivos de trabajo cuando aparezca publicada una versión de ISO/IEC 25010.

3 Redes Bayesianas

El modelo que vamos a construir tiene en cuenta los problemas usuales que se presentan en la construcción de modelos en la Ingeniería del Software: la recogida de datos necesarios para llevar a cabo el estudio (resultados previos, juicios expertos); estimación de los parámetros intrínsecos del modelo; y datos fiables que nos permitan generalizar las conclusiones que obtengamos.

El primer problema es determinar qué método o herramienta estadística utilizar para realizar un estudio a posteriori entre la Calidad en Uso y la Calidad Externa. Las técnicas clásicas, como el Análisis de Componentes Principales o Modelos Lineales de Regresión, no son las apropiadas para este estudio, pues realizan un análisis clásico,"hacia adelante", y no a posteriori como nosotros deseamos; también es cierto que los métodos clásicos necesitan una gran cantidad de datos para que dichos análisis sean eficientes y fiables, cosa que no es muy habitual en la construcción de modelos para la Ingeniería del Software.

El siguiente problema es cómo plantear el modelo que explique de forma clara las relaciones e influencias entre las características de la Calidad Externa con las características de la Calidad en Uso. Estas relaciones se pueden plasmar en un grafo dirigido causal.

Por último, hay que decidir cómo medir las relaciones de influencia que se han establecido en el grafo. Ya que los nodos de dicho grafo se pueden considerar variables aleatorias parece que la medida más lógica es medir la influencia de unos nodos sobre otros en términos de probabilidad.

Las Redes Bayesianas se postulan como la mejor herramienta para realizar este estudio. Una Red Bayesiana es un grafo acíclico y dirigido, cuyos nodos son variables de incertidumbre (variables aleatorias) y sus arcos representan la influencia causal entre los nodos. Para cada nodo hay asociada una tabla de probabilidades condicionadas, que explican la influencia causal entre los nodos [7].

Para definir una red bayesiana necesitamos:

1. Dar un conjunto de variables aleatorias (nodos) y las relaciones (influencia causal) entre dichas variables.
2. Construir un grafo con los nodos y las relaciones establecidas entre ellos.
3. Definir tablas de probabilidad condicionada asociadas a cada nodo. Estas tablas determinan el grado de influencia de los nexos del grafo y se utilizan para calcular la distribución de probabilidad de cada nodo en la Red Bayesiana.

A partir de las redes bayesianas podemos: representar las diferentes relaciones entre las características y subcaracterísticas de la Calidad Externa y la Calidad en Uso y el grado de dependencia o influencia entre ellas, incorporando la subjetividad en los criterios de evaluación y la incertidumbre que aparece cuando se establece un criterio

de decisión que engloba a varios criterios de decisión ya establecidos; por ejemplo: los criterios basados en juicios de expertos.

La red así definida, junto con los algoritmos bayesianos de propagación de probabilidades, nos permite hacer inferencia sobre los valores de los nodos de la red a partir de un conjunto de observaciones de evidencias disponibles. Esta inferencia dependerá del objetivo del estudio: o determinar el conjunto de subcaracterísticas de la Calidad Externa que debemos considerar para garantizar un nivel requerido de la Calidad en Uso; o determinar la probabilidad de obtener ciertos resultados en el futuro (inferencia predictiva).

4 Construcción de una red bayesiana para la Calidad en Uso

Nuestro trabajo se basa en la hipótesis de que la Calidad Externa tiene influencia en la Calidad en Uso, que esta influencia se puede modelar mediante una red bayesiana y que esta red nos sirve para realizar un estudio a posteriori sobre el nivel requerido de la Calidad en Externa para asegurar un nivel deseado de la Calidad en Uso.

La forma de modelar las relaciones entre las características de la Calidad Externa y sus subcaracterísticas usando redes bayesianas se muestran en la Figura 2, en la que los nodos de la red son las características y subcaracterísticas, y los nexos representan las relaciones entre ellas.

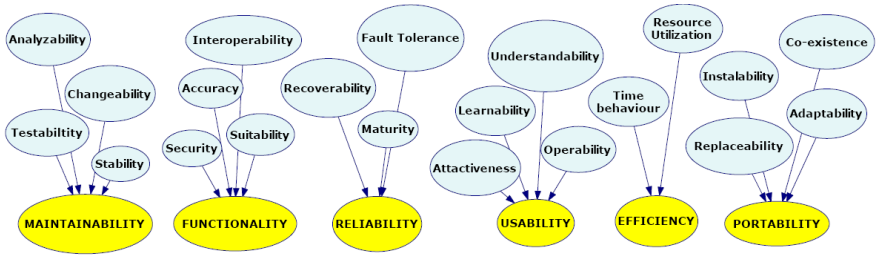


Fig. 2. Características y subcaracterísticas de la Calidad Externa (ISO/IEC 9126)

En segundo lugar modelamos la relación entre la Calidad en Uso y sus cuatro características mediante redes bayesianas. La Figura 3 muestra la red para la Calidad en Uso; hay que recordar ISO/IEC 9126 no define subcaracterísticas para ella.

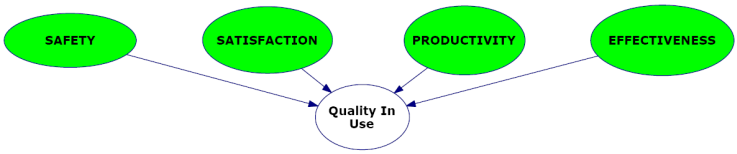


Fig. 3. Características de la Calidad en Uso (ISO/IEC 9126)

En tercer lugar, necesitamos unir, mediante nexos, estas dos redes para poder definir las relaciones que estamos buscando. Las posibles formas en la que estas dos redes se pueden modelar se describen en un trabajo previo [9], en el que se discuten las distintas opciones para definir las relaciones entre los nodos de las redes de la Calidad Externa y la Calidad en Uso. Básicamente es preciso definir las relaciones entre las subcaracterísticas de la Calidad Externa y las características de la Calidad en

Uso. Estas relaciones determinarán el grado de influencia de cada subcaracterística de la Calidad Externa sobre cada característica de la Calidad en Uso.

Evidentemente se podrían haber definido directamente las relaciones entre los nodos que representan a las características de la Calidad Externa (Mantenibilidad, Funcionalidad, etc.) y los nodos que representan a las características de la Calidad en Uso (Seguridad, Satisfacción, etc.). Pero si hiciésemos esto nos encontraríamos que si una característica de la Calidad Externa está relacionada con una característica de la Calidad en Uso, todas sus subcaracterísticas tendrán algún grado de influencia sobre esa característica de la Calidad en Uso. Además, cada subcaracterística de la Calidad Externa tendrá el mismo grado de influencia sobre una característica de la Calidad en Uso dada, lo que no es correcto.

El enfoque que parece más lógico es relacionar cada subcaracterística de la Calidad Externa con las características de la Calidad en Uso, pero esto produce un número alto de entradas sobre los nodos que representan a las características de la Calidad en Uso. Por este motivo, introducimos nodos sintéticos, práctica común por otra parte, con el fin de simplificar el número de relaciones en la red bayesiana final. Estos nodos se definen, de forma natural, como nodos intermedios entre las subcaracterísticas de cada característica de la Calidad Externa y las características de la Calidad en Uso. Así cada nodo sintético recoge la influencia particular de una característica de la Calidad Externa sobre una característica de la Calidad en Uso.

Como desventaja, el concepto de características de Calidad Externa se pierde. Este inconveniente puede obviarse si hacemos uso de las propiedades y operaciones de las redes bayesianas. De hecho, lo que hacemos es crear cuatro redes individuales, una para cada característica de la Calidad en Uso, y unirlas para crear una red completa. Estas cuatro redes son las que constituyen el modelo básico de nuestra propuesta. Una de ellas, la correspondiente a la característica Satisfacción de la Calidad en Uso, es la que se muestra en la Fig. 4.

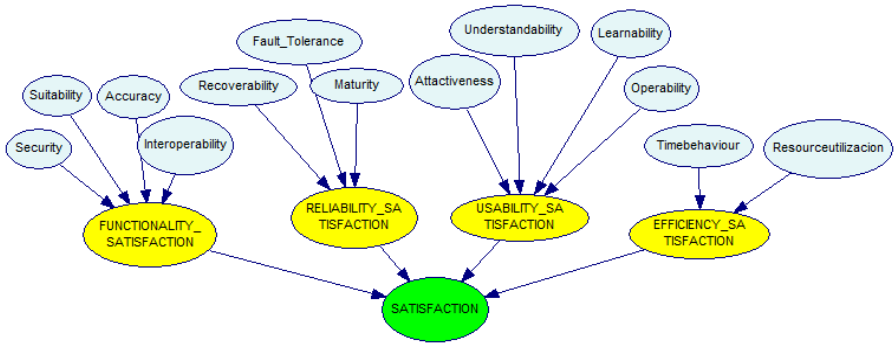


Fig. 4. Red Bayesiana para la característica de la Calidad en Uso *Satisfacción*.

Obsérvese que existen tres niveles de nodos: las subcaracterísticas de la Calidad Externa, primer nivel (nodos padres), las características de la Calidad Externa, nivel intermedio y la característica de la Calidad en Uso particular que se esté estudiando, último nivel.

Este modelo permite el estudio de cada característica de la Calidad en Uso de forma independiente y observar de forma más fácil la influencia de las subcaracterísticas de la Calidad en Externa sobre cada característica de la Calidad en

Uso. Una vez que las cuatro redes están construidas se pueden combinar para hacer un estudio global de la Calidad en Uso.

El último paso para la construcción de la red sería definir los estados o valores de cada nodo de la red y de las tablas de probabilidad asociada a cada uno de ellos. En estas tablas se reflejan las probabilidades de los estados de un nodo en función de los estados de los nodos anteriores que influyen sobre él. Un caso detallado para los componentes software se desarrolla en el siguiente apartado.

5 Uso las Redes Bayesianas para evaluar la calidad de los Componentes Software

Siguiendo los pasos indicados en el punto anterior, hemos definido y construido cuatro Redes Bayesianas, una para cada característica de la Calidad en Uso, para un dominio de aplicación específico, en nuestro caso los componentes software. Estas redes pueden ser una herramienta útil para razonar sobre la relación e influencia de la Calidad Externa sobre la Calidad en Uso en este dominio de aplicación, que es uno de los objetivos de nuestro trabajo.

5.1 Dominio de aplicación

En este punto es muy importante considerar cuál es el dominio de aplicación para el que tenemos que construir las Redes Bayesianas, ya que la mayor o menor relevancia de cada subcaracterística, así como su influencia sobre las características de la Calidad en Uso, dependerán en gran medida de dicho dominio de aplicación y del contexto de uso particular.

En nuestro estudio utilizamos los componentes software de una compañía de software que produce, vende y mantiene aplicaciones para construcción y arquitectura. El contexto de uso ha sido el desarrollo de estas aplicaciones utilizando los componentes seleccionados; es decir, los usuarios fueron los desarrolladores de sistemas cuya tarea es integrar los componentes de software para construir las aplicaciones que la compañía vende y mantiene.

Es importante señalar que es esencial adaptar el modelo de calidad ISO a las necesidades e intereses particulares del dominio de aplicación, con el fin de determinar si algunas de las características no son relevantes para los productos de ese dominio. Así, el caso particular del dominio de aplicación del CBSD, dos de las características de la Calidad Externa se han omitido: *Mantenibilidad* y *Portabilidad*. Estas dos características no eran relevantes para los usuarios finales de los componentes de software en sus contextos de uso, porque son integradores de sistemas. Esta es la razón por la que no aparecen en la red se muestra en la Fig. 5. Para el resto de las características que aparecen en la red, su grado de influencia sobre la Calidad en Uso será calculado por las Redes Bayesianas propuestas.

5.2 Tablas de probabilidad

Para definir los estados de los nodos utilizamos inicialmente las categorías definidas en el estándar internacional 1061 de IEEE[4]: Aceptable, Marginal e Inaceptable. Sin embargo, al hacer la adaptación al contexto de los componentes software,

descubrimos que trabajando con las categorías de Aceptable e Inaceptable era suficiente. De acuerdo al estándar, el valor Marginal se usa sólo cuando la característica no puede ser asignada de forma clara a los valores de Aceptable o Inaceptable. Así, un valor Marginal sólo debe representar el error permisible cuando evaluamos una característica o cuando tenemos incertidumbre sobre ella.

Las tablas de probabilidad de los nodos iniciales son probabilidades a priori derivadas de la evaluación de las subcaracterísticas de la Calidad Externa. En el caso de nuestro estudio hemos preguntado a los usuarios sobre todas las subcaracterísticas de la Calidad Externa, incluso aquellas para las que disponíamos de indicadores objetivos y validados, con el fin de usar estas respuestas como preguntas de control.

Las tablas de probabilidad de los nodos correspondientes a las características de la Calidad Externa, se han definido teniendo en cuenta el grado de influencia (mediante pesos) que cada una de ellas tiene sobre la correspondiente característica de la Calidad en Uso a la que se refiere el estudio. Los nodos del tercer nivel representan a las características de la Calidad en Uso y se construyen de forma similar a las anteriores.

Los valores de las tablas de probabilidad de los nodos del segundo y tercer nivel de la red dependen del dominio de aplicación y del objetivo de los usuarios. Para calcularlas preguntamos a los usuarios, mediante cuestionarios, su opinión subjetiva sobre un conjunto de componentes software que utilizan en la construcción de sus aplicaciones software. Por un lado, pedíamos a los usuarios que valorasen, como Aceptable o Inaceptable, todas las subcaracterísticas y las características de la Calidad Externa. Por otro lado tenían que valorar la característica de la Calidad en Uso objeto del estudio, dando un porcentaje de cómo de Aceptable era el componente respecto a dicha característica.

Con las respuestas obtenidas analizamos, en primer lugar, las posibles observaciones anómalas o respuestas inconsistentes de los encuestados que se debían tener en cuenta a la hora de obtener los parámetros de la red, ya que las tablas de probabilidad heredaban algunas de esas inconsistencias.

Esto nos proporcionó una primera versión de las tablas de probabilidad asociada a cada nodo que tuvieron que ser refinadas para evitar situaciones en las que las respuestas no eran consistentes entre sí. El juicio del experto fue fundamental para este propósito.

5.3 Validación del modelo

Después de la construcción y definición de las tablas de probabilidad de la red obtuvimos cuatro redes (cada una relativa a una de las características de la Calidad en Uso). Para validar que el modelo de decisión representado por la red es correcto, utilizamos un conjunto de datos controlados y proporcionados por los usuarios que incluía observaciones detectadas como anómalas. Evaluamos la red con todos ellos y obtuvimos el la probabilidad, expresada en porcentaje, de Aceptable del nodo final para cada usuario. Teníamos así, para cada dato, el porcentaje de Aceptable para la característica de la Calidad de Uso dado por el encuestado y el porcentaje de Aceptable para dicha característica calculada por la red. Comprobamos si dichos porcentajes estaban correlados obteniendo que el Coeficiente de Determinación, R^2, fluctuaba entre 0.81, cuando se incluían las observaciones anómalas en el conjunto de datos controlados y 0.9 cuando el conjunto de datos no incluían los datos anómalos. La red parecía adecuada para el análisis deseado.

5.4 Resultados para los componentes software

Por motivos de espacio, vamos a detallar los resultados obtenidos en una de las redes construidas, en concreto para la Efectividad. La red de la Fig. 5 se centra en las cuatro características (y sus subcaracterísticas) de la Calidad Externa que tienen mayor relevancia en el contexto de los componentes.

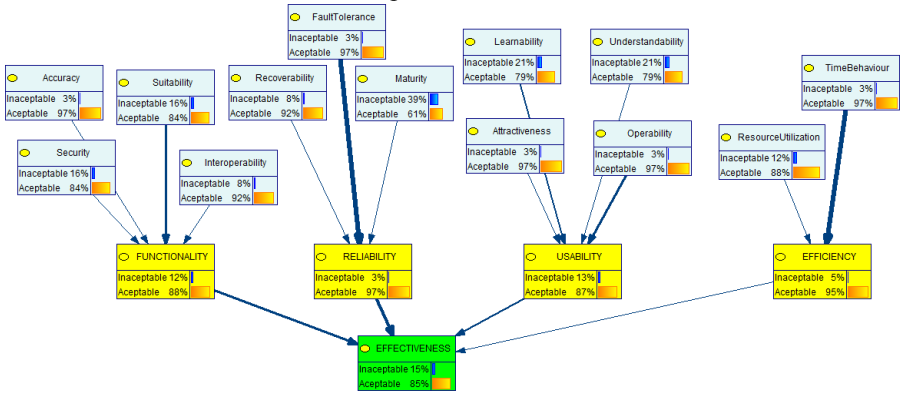


Fig. 5. Diagrama de Influencia para la Efectividad (diagrama de barras).

Instanciando el nodo de Efectividad a Aceptable se calculan las probabilidades a posteriori para los valores de Aceptable e Inaceptable de los nodos de la red y en particular para las subcaracterísticas de la Calidad Externa.

El análisis a posteriori que se realiza nos permite conocer las subcaracterísticas con mayor influencia sobre la Efectividad. Sin embargo, como la probabilidad que tiene la Efectividad de ser Aceptable inicialmente sin instanciar ningún nodo es alta (85%), los incrementos que en la Efectividad produce instanciar una única subcaracterística no sobrepasan el 2%. Por tanto, es muy difícil clasificar o determinar qué subcaracterísticas, de forma individual, tienen una influencia positiva determinante.

Lo que sí se ha observado es que, en general, si tienen una influencia positiva (aumenta la probabilidad de ser Aceptable) también la tienen negativa (disminuye la probabilidad de ser Aceptable). Las variaciones negativas sí son de mayor envergadura y más fáciles de apreciar.

En cualquier caso las variaciones tanto positivas como negativas se hacen más evidentes instanciando un conjunto de subcaracterísticas que influyan de forma significativa sobre la Efectividad.

5.4.1 Clasificación de las subcaracterísticas. Análisis hacia atrás.

Para clasificar las subcaracterísticas de la Calidad Externa hemos utilizado la red bayesiana observando las probabilidades a priori de sus nodos cuando la red no presenta evidencias, y cuando se evidencia el nodo final de Efectividad tanto como Aceptable como Inaceptable.

Hemos analizado las variaciones que experimentan los porcentajes de ser Aceptable de cada subcaracterística cuyos resultados se muestran en la Tabla 1. La columna Vef(A) indica la variación entre el porcentaje de ser Aceptable, inicialmente, de cada subcaracterística (**$P(Sc_i$=Aceptable)**) y el porcentaje de ser Aceptable cuando la

Efectividad es Aceptable (**P(Sci=Aceptable | Eff=Aceptable)**); la columna Vef(In) representa la variación que experimenta el porcentaje de ser Aceptable, inicialmente, cada subcaracterística (**P(Sci=Aceptable)**) y el porcentaje cuando la Efectividad es Inaceptable (**P(Sci=Aceptable | Eff=Inaceptable)**).

Subcaracterística Sc_i	**P(Sc_i=A)**	**P(Sc_i=A | Eff=A)**	**P(Sc_i=A | Eff =In)**	**Vef(A)**	**Vef(In)**
Adecuación	84%	86%	70%	2%	14%
Seguridad	84%	85%	76%	1%	8%
Aprendibilidad	79%	80%	72%	1%	7%
Tolerancia Fallos	97%	98%	91%	1%	6%
Comprensibilidad	79%	80%	74%	1%	5%
Recuperabilidad	92%	93%	89%	1%	3%
Operabilidad	97%	97%	95%	0%	2%
Corrección	97%	97%	95%	0%	2%
Interoperabilidad	92%	92%	91%	0%	1%
Madurez	61%	61%	62%	0%	1%
Comp. Temporal	97%	97%	96%	0%	1%
Atractividad	97%	97%	97%	0%	0%

Tabla 1. Resultado de evidenciar la Efectividad como Aceptable e Inaceptable.

Se puede apreciar que hay un grupo de subcaracterísticas que presentan mayores variaciones en sus porcentajes que denominaremos Esenciales. En este grupo podemos incluir las subcaracterísticas de Adecuación, Seguridad, Aprendibilidad, Tolerancia a Fallos y Comprensibilidad. También se puede observar que hay otro grupo, que denominaremos No Significativas, cuyas variaciones son poco significativas y donde se que incluye al resto de subcaracterísticas.

Así mediante el uso de la red hacia atrás, obtenemos una primera clasificación de las subcaracterísticas de la Calidad Externa en dos categorías: Esenciales y No Significativas.

5.4.2 Clasificación de las subcaracterísticas. Análisis *hacia adelante*.

Utilizar únicamente el criterio anterior para clasificar las subcaracterísticas de Calidad Externa no es suficiente. Por ello, hemos realizado un análisis de la red hacia delante, cuantificando la influencia que tienen las subcaracterísticas sobre el valor de Aceptable para la Efectividad, de tal modo que podamos tener un refinamiento sobre la clasificación propuesta.

La Tabla 2 muestra el porcentaje del valor Aceptable para la Efectividad cuando la red no muestra ninguna evidencia (**P(Eff=A)**) que es el 85%; cuando se evidencia cada subcaracterística como Aceptable (**P(Eff = A | Sc_i = A)**); cuando se evidencia cada subcaracterística como Inaceptable (**P(Eff = A | Sc_i = In)**); y las variaciones de los porcentajes en cada caso (Vsc(A), Vsc(In)), siendo:

$$Vsc(A) = P(Eff = A \mid Sc_i = A) - P(Eff=A) \quad y$$

$$Vsc(In) = P(Eff=A) - P(Eff = A \mid Sc_i = In)$$

En este caso podemos observar que existen tres grupos de subcaracterísticas. El primero formado por las subcaracterísticas *Adecuación y Tolerancia a Fallos* que influyen claramente sobre la Efectividad. Estas dos subcaracterísticas aparecían también en la clasificación anterior en el grupo de **Esenciales**.

En sentido opuesto, hay claramente otro grupo con una influencia muy poco significativa sobre la Efectividad que estaría formado por las subcaracterísticas *Interoperabilidad, Madurez, Atractividad, Utilización de Recursos, Comportamiento Temporal*, que también aparecían en la clasificación anterior en el grupo de **No Significativas**.

Subcaracterística Sc_i	**P(Eff=A)**	**P(Eff = A | Sc_i = A)**	**P(Eff =A | Sc_i = In)**	**Vsc(A)**	**Vsc(In)**
Tolerancia Fallos	85%	86%	53%	1%	32%
Adecuación	85%	87%	72%	2%	13%
Seguridad	85%	86%	77%	1%	8%
Corrección	85%	85%	76%	0%	9%
Operabilidad	85%	85%	76%	0%	9%
Aprendibilidad	85%	86%	79%	1%	6%
Recuperabilidad	85%	85%	79%	0%	6%
Comprensibilidad	85%	86%	81%	1%	4%
Comp. Temp	85%	85%	80%	0%	5%
Atractividad	85%	85%	82%	0%	3%
Interoperabilidad	85%	85%	83%	0%	2%
Madurez	85%	85%	84%	0%	1%
Util. Recursos	85%	85%	84%	0%	1%

Tabla 2. Resultado de evidenciar las subcaracterísticas como Aceptable e Inaceptable.

Queda un grupo intermedio de subcaracterísticas que no podemos clasificar en ninguno de los dos grupos anteriores.

Por un lado las subcaracterísticas Seguridad, Aprendibilidad y Comprensibilidad, que aparecían como Esenciales, no parecen ser tan influyentes sobre la Efectividad cuando se hace el análisis hacia delante. El decremento para Inaceptable (V(In)) para ellas, es mucho menor (8%, 6% y 4% respectivamente) que el decremento para Adecuación y Tolerancia a Fallos (13% y 32%, respectivamente).

Por otro lado, las subcaracterísticas Corrección, Operabilidad y Recuperabilidad, que aparecían como No Significativas, cuando se realiza el análisis hacia delante, sí parecen tener mayor influencia que las del resto del grupo No Significativas. El decremento para Inaceptable en la Efectividad (9%, 9% y 6% respectivamente) es mayor que el que se observa para el resto de subcaracterísticas del grupo de No Significativas.

Como vemos, tenemos un grupo intermedio de subcaracterísticas que es difícil clasificar y sobre todo valorar su influencia sobre la Efectividad. Hemos decido subdividir dicho grupo en dos categorías, según tengan una influencia media, que denominaremos Secundarias o una influencia baja, que denominaremos Auxiliares atendiendo fundamentalmente a la variación que producen en la Efectividad cuando se evidencian como Inaceptable.

Las **Secundarias,** que tienen una influencia "media", son las subcaracterísticas *Corrección*, *Seguridad* y *Operabilidad.* Las **Auxiliares,** que tienen una influencia baja, son las subcaracterísticas: *Recuperabilidad, Comprensibilidad* y *Aprendibilidad*,

5.4.3 Clasificación de las subcaracterísticas utilizando los dos análisis.

Utilizando los dos análisis y el conjunto de condiciones propuestos en los puntos anteriores, las subcaracterísticas de Calidad Externa quedan clasificadas en cuatro categorías como se resume en la Tabla 3.

Categoría	Subcaracterísticas de la Calidad Externa
C1: Esenciales (Influencia Alta)	Adecuación, Tolerancia a Fallos.
C2: Secundarias (Influencia Media)	Corrección, Seguridad, Operabilidad.
C3: Auxiliares (Influencia Baja)	Recuperabilidad, Aprendibilidad, Comprensibilidad
C4: No Significativas (Influencia Muy baja o Nula)	Interoperabilidad, Madurez, Atractividad, Utilización de Recursos, Comportamiento Temporal.

Tabla 3. Clasificación de las subcaracterísticas de Calidad Externa según su influencia.

En la Tabla 4 se detallan todas las condiciones, recogidas en los dos análisis en términos de variaciones de los porcentajes. Para clasificar una subcaracterística como Esencial o No Significativa debe cumplir las cuatro condiciones correspondientes. En caso de no cumplir alguna de ellas, se clasifica como Secundaria o Auxiliar atendiendo a la valor de Vsc(In).

Categoría	Vef(A)	Vef(In)	Vsc(A)	Vsc(In)
Esenciales	≥ 1%	≥ 5%	≥ 1%	≥ 10%
Secundarias	No cumplen todas condiciones de Esenciales ni de No Significativas			≥ 8%
Auxiliares	No cumplen todas condiciones de Esenciales ni de No Significativas			< 8%
No Significativas	< 1%	< 5%	< 1%	≤ 5%

Tabla 4. Condiciones para la clasificación de las subcaracterísticas de Calidad Externa

5.4.4 Valoración de los resultados según categorías.

Como hemos indicado anteriormente, las variaciones en los porcentajes de los valores de Aceptable o Inaceptable para la Efectividad y, por tanto, la influencia de las distintas subcaracterísticas de la Calidad Externa, quedan de forma más evidente cuando las subcaracterísticas que pertenecen a una misma categoría toman el mismo valor.

La Tabla 5 muestra la influencia de cada categoría cuando todas sus subcaracterísticas toman el mismo valor (Aceptable o a Inaceptable), recogiendo el porcentaje de Aceptable para el nodo de Efectividad. Para ilustrar la influencia de cada categoría hemos construido la Tabla 5, donde todas las subcaracterísticas de una categoría toman el mismo valor (se evidencia todas a Aceptable o a Inaceptable) recogiendo el valor para la probabilidad de Aceptable de la Efectividad.

Categoría	P(Eff = A | C_i = A)	P(Eff = A | C_i = In)
C1: Esenciales (2 subcaract.)	88%	42%
C2: Secundarias (3 subcarcat.)	87%	60%
C3: Auxiliares (3 subcarcat.)	88%	70%
C4: No Significativas (5 subcarcat.)	85%	76%

Tabla 5. Porcentajes de Aceptable para Efectividad según se evidencian las categorías

Estos resultados confirman lo esperado: la influencia de la categoría Esenciales, con sólo dos subcaracterísticas, se hace notar en el porcentaje de Aceptable para Efectividad. Cuando se instancian las dos subcaracterísticas (Esenciales) como Inaceptable, el porcentaje de Aceptable de la Efectividad baja al 42%.

Por otro lado, la influencia del grupo No Significativas es muy baja: cuando sus cinco subcaracterísticas se instancian como Inaceptable la probabilidad de Aceptable de la Efectividad apenas disminuye al 76%, que es sustancialmente menor al caso de instanciar las dos subcaracterísticas Esenciales (42%) o las tres Secundarias (60%).

Si esta prueba la extendemos a unir e instanciar simultáneamente con el mismo valor, por un lado a las 5 subcaracterísticas Esenciales y Secundarias y por otro lado, las 9 subcaracterísticas No Significativas y Auxiliares obtenemos los resultados mostrados en la Tabla 6, que parecen confirmar la clasificación hecha.

Grupo	**P(Eff = A)**	**P(Eff = A|G_i = A)**	**P(Eff = A|G_i = In)**
G1: {Esenciales} ∪ {Secundarias} 5 subcaracterísticas	85%	90%	24%
G2: {No Significativas} ∪ {Auxiliares} 8 subcaracterísticas	85%	88%	67%

Tabla 6. Valores para la Efectividad según se evidencien los grupos de subcaracterísticas

Este resultado nos indica qué conjunto de subcaracterísticas debemos vigilar en el desarrollo de nuestro producto (Esenciales y Secundarias), y qué características podemos prestar menor atención (Auxiliares y No Significativas).

6 Conclusiones

En este trabajo hemos mostrado cómo construir un Red Bayesiana para determinar la influencia de las subcaracterísticas de Calidad Externa sobre la Calidad en Uso de un producto software, y cómo esta red se puede usar para determinar las características de Calidad Externa que realmente importan a los usuarios en su particular contexto de uso. Hay varias aplicaciones interesantes de nuestra contribución. Primera, las Redes Bayesianas pueden ser una herramienta muy eficiente para el personal de marketing que trabaja en compañías que diseñan y venden productos software. Las Redes Bayesianas permiten identificar las características del producto que no son realmente relevantes para su público objetivo y cuyos costes de mantenimiento y desarrollo asociados se pueden reducir drásticamente. Segunda, una Red Bayesiana de este tipo puede ayudar a las compañías interesadas en seleccionar y utilizar productos software, a identificar cuáles son las características del producto que realmente les interesan. A partir de estas premisas, se buscarán los productos que destaquen en esas características, poniendo menos atención en otras propiedades que no son relevantes para su contexto de uso. En otras palabras, nuestro trabajo puede ser utilizado de forma efectiva por los clientes para la selección de los productos que mejor se ajusten a sus necesidades con el menor coste posible.

Por supuesto, el nivel de influencia de las subcaracterísticas de la Calidad Externa sobre las Características de la Calidad en Uso puede variar entre diferentes dominios de aplicación. Las relaciones modeladas por las Redes Bayesianas pueden necesitar

adaptarse a otros dominios. En cualquier caso, el método presentado en este documento sigue siendo válido.

Hay varias actividades que tenemos previsto abordar a corto plazo. En primer lugar, queremos refinar esta propuesta combinándola con la aplicación de técnicas bayesianas robustas (Modelo de Regresión Dicotómico Robusto) durante la definición inicial de la Red Bayesiana y de las tablas de probabilidades condicionadas, a partir del cual podremos confirmar (y mejorar) las relaciones definidas en la Fig. 5 entre subcaracterísticas de Calidad Externa y las características de Calidad en Uso.

Por último, esperamos proporcionar una aportación útil al Grupo de Trabajo de ISO que define la nueva familia de normas SQuaRe basada en nuestra investigación, no sólo confirmando la existencia de la influencia de Calidad Externa sobre la Calidad en Uso, sino además cuantificándola en algunos contextos de uso concretos; y, lo más importante, destacando el papel primordial que desempeña la Calidad en Uso en la evaluación de la calidad de cualquier producto de software como motor del resto de puntos de vista de la calidad.

Referencias

1. Bertoa, M.F., Troya, J.M., and Vallecillo, A., Measuring the usability of software components. Journal of Systems and Software. 79(3): 427-439 (2006)
2. Botella, P., Burgués, X., Carvallo, J.P., Franch, X., Pastor, J.A., and Quer, C., Towards a Quality Model for the Selection of ERP Systems. Component-Based Software Quality. Springer-Verlag: 225-245 (2003)
3. Dromey, R.G., A Model for Software Product Quality. IEEE Transaction on Software Engineering. 21(2): 146-162 (1995)
4. IEEE, Std. 1061-1998. IEEE Standard for a Software Quality Metrics Methodology. (1998).
5. ISO/IEC, ISO/IEC 9126. Software Engineering-Product Quality. Parts 1 to 4. International Organization for Standardization/International Electrotechnical Commission. (2001).
6. Jagdish, B., A Hierarchical Model for object-oriented Design Quality Assessment. IEEE Transaction on Software Engineering. 28(1): 4-17 (2002)
7. Jensen, F.V., Bayesian Networks and Decisions Graphs. Springer-Verlag. (2001).
8. Liu, K., Zhou, S., and Yang, H. Quality Metrics of Object Oriented Design for Software Development and Re-Development. In: The First Asia-Pacific Conference on Quality Software (APAQS'00) pp. 127-135. (2000)
9. Moraga, M.Á., Bertoa, M.F., Morcillo, M.C., Calero, C., and Vallecillo, A. Evaluating Quality-in-Use Using Bayesian Networks. In: 12th Workshop on Quantitative Approaches on Object Oriented Software Engineering (QAOOSE 2008). Paphos, Cyprus. (2008)
10. Moraga, M.Á., Calero, C., and Piattini, M., Comparing different quality models for portals. Online Information Review. 30(5): 555-568 (2006)
11. Neil, M., Krause, P., and Fenton, N.E., Software Quality Prediction Using Bayesian Networks, in Software Engineering with Computational Intelligence. Chapter 6. 2003, Kluwer

Validación empírica de medidas para procesos ETL en almacenes de datos

Lilia Muñoz[1], Jesús Pardillo[2], Jose-Norberto Mazón[2], and Juan Trujillo[2]

[1] Grupo de Investigación Lucentia. Departamento de Sistemas de Información, Control y Evaluación de Recursos Informáticos
Universidad Tecnológica de Panamá, Panamá
lilia.munoz@utp.ac.pa
[2] Grupo de Investigación Lucentia, Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, España
{jesuspv,jnmazon,jtrujillo}@dlsi.ua.es

Resumen En un almacén de datos, los procesos ETL (*extraction, transformation, load*) se encargan de la extracción de datos de las fuentes de origen que puede contener el almacén de datos. Debido a su relevancia, la calidad de estos procesos debe ser evaluada formalmente, desde las primeras etapas de desarrollo. A fin de evitar tomar decisiones erradas como resultado de datos incorrectos. En este trabajo, se presenta un conjunto de medidas para evaluar la complejidad estructural de los modelos de procesos ETL a nivel conceptual. Además, este estudio es acompañado por cuatro experimentos, cuyo objetivo es la validación empírica de las medidas propuestas. La principal ventaja de este enfoque es la evalua-ción temprana de los modelos de procesos ETL, lo cual permite a los diseñadores realizar las tareas de mantenimiento de manera más fácil. Esta propuesta se basa en diagramas de actividad UML (*Unifield Modeling Language*) para modelar procesos ETL y en el marco de modelado de procesos FMESP (*Framework for the Modeling and Evaluation of Software Processes*).

Key words: Procesos ETL, validación, medidas, almacenes de datos

1. Introducción

En los años '90s, Inmon [9] definió el término Almacén de Datos (AD) como: *una colección de datos orientados por temas, integrados, variables en el tiempo y no volátiles para el apoyo de la toma de decisiones*. Un AD es "integrado", porque los datos que se introducen en el almacén se obtienen de una variedad de fuentes de datos (sistemas heredados, bases de datos relacionales, ficheros COBOL, etc.). Para lograr la integración de esa variedad de fuentes, se utilizan los procesos ETL. Dichos procesos son los responsables de la extracción de los datos a partir de las diversas fuentes de datos heterogéneas, de la transformación de estos (conversión, limpieza, etc.), y de su carga en el AD. Se reconoce ampliamente que el diseño y mantenimiento de los procesos ETL son factores claves en el éxito de proyectos de AD [25].

Por su parte, un proceso ETL es extremadamente complejo, propenso a errores y consume mucho tiempo [24]. Se ha argumentado ampliamente, en la literatura, que los

procesos ETL son costosos y que son una de las partes más importantes del desarrollo de un AD [9,27]. En [22], se reporta que los costos de herramientas ETL se estiman en al menos la tercera parte de los gastos del presupuesto de un AD. Por estas razones, así como por el alto coste de adquisición y mantenimiento, muchas organizaciones prefieren desarrollar sus propios procesos ETL.

En los últimos años se han definido varias propuestas para el modelado conceptual y lógico de procesos ETL para AD [13,23,26,27,28]. Dada la gran complejidad de los procesos ETL, estas propuestas tratan sobre el diseño, mantenimiento y gestión de los procesos ETL. Por otro lado, un mal diseño de los procesos ETL, puede suponer errores graves en la demora de la carga en el AD. Por su parte, los esfuerzos en obtener la calidad de los ADs se han orientado a la calidad del producto final, no así a la calidad del modelado de procesos ETL que es parte fundamental del mismo. En la literatura más relevante sobre este tema se han encontrado algunas aproximaciones [1,20,28,29] donde se han presentado medidas de calidad para procesos ETL. Sin embargo, en ninguna de estas aproximaciones se han validado formal o empíricamente las medidas propuestas.

Con el fin de cubrir esta carencia, en este artículo se describe un conjunto de medidas para evaluar la mantenibilidad del modelado conceptual de procesos ETL, partiendo de la hipótesis de que la baja mantenibilidad (facilidad de mantenimiento) de un modelo de procesos ETL influye negativamente en su calidad, y por lo tanto, puede tener repercusión en el desarrollo de AD (más costosos en recursos y tiempo). Las medidas son aplicables a una aproximación de modelado conceptual de procesos ETL basada en diagramas de actividad UML [13]. Mediante la utilización de diagramas de actividad UML conseguimos una mejor representación de los procesos ETL, permitiendo representar los aspectos dinámicos de dichos procesos. Por su parte, la propuesta para modelado conceptual de procesos ETL presentada en [13], se enmarca dentro de una propuesta global para acometer el desarrollo de AD con MDA [15], en este marco de trabajo se han desarrollado las diferentes capas de la arquitectura de un AD con MDA [12,17], lo que ha permitido el desarrollo del AD de manera sistemática y automatizada.

Para la definición de las medidas se utilizó el marco FMESP (*Framework for the Modeling and Evaluation of Software Processes*) [7]. FMESP consiste en un marco de modelado y medición del proceso del software, en el que los modelos se representan en base a SPEM (*Software Process Engineering Metamodel Specification*) [14]. Sin embargo, dada su generalidad y flexibilidad lo hemos adaptado para la evaluación de modelos de procesos ETL a nivel conceptual. El resto del artículo se estructura como sigue: En la sección 2 se describe el trabajo relacionado. Las medidas propuestas definidas para modelos de procesos a nivel conceptual se presentan en la sección 3 y a continuación en la sección 4 se describe el proceso experimental para el desarrollo de los experimentos. Finalmente, en la sección 5 se aportan las conclusiones y trabajo futuro.

2. Trabajo relacionado

La calidad de un AD vendrá influenciada por la calidad del SGBD (*Sistema Gestor de Bases de Datos*), por la calidad de los modelos de datos y por la calidad de los procesos ETL [19]. El trabajo relacionado lo enmarcamos en estos dos últimos aspectos.

Calidad para modelos de datos. Con respecto a las medidas de calidad para modelos de datos podemos hacer referencia a los trabajos presentados en [4,16,18]. Estas propuestas son buenas aproximaciones a las medidas de ADs; sin embargo, no son completas, ya que no son parte de un modelo de calidad que permita a los diseñadores su uso de una forma sistemática y objetiva. Por otro lado, podemos encontrar la propuesta de un modelo de calidad de un AD, llamada DWQ (*Data Warehouse Quality*) [10] que intenta asegurar la calidad de los datos almacenados para mejorar el uso del AD; esta aproximación evalúa algunas dimensiones de calidad, centrándose en la calidad de los datos. No obstante, la arquitectura de un AD es compleja, por lo que consideramos que la calidad de los ADs debe abarcar todas las capas que componen el almacén, y en ninguna de estas propuestas están considerados los aspectos de calidad de procesos ETL.

Calidad para procesos ETL. En la literatura consultada existen algunas aproximaciones sobre medidas de calidad para procesos ETL. En [28], los autores modelan un escenario ETL como un gráfico e introducen la importancia de las medidas en los nodos del gráfico. Los mismos autores definen, en [29] una colección de medidas, usadas para evaluar los flujos de trabajo de los procesos ETL. Por su parte, en el campo de la integración de datos para procesos ETL, las medidas se pueden ubicar en las siguientes categorías [20]: tipos de datos, conformidad del domino de datos, características estadísticas de un conjunto de datos (valor máximo, valor mínimo, etc.) y relaciones referenciales. En [1], se presenta una propuesta para verificar la consistencia de los datos que son cargados en el AD. Se calculan la *Entropía de Shannon* [21] en particiones producidas por un conjunto de atributos, demostrando que estos valores pueden ser utilizados como señales de problemas en el proceso de extracción de los datos. Sin embargo, en ninguna de estas aproximaciones se han validado formal y empíricamente las medidas propuestas, de tal manera que podamos contar con parámetros objetivos para elegir entre ciertos modelos de procesos ETL. Por lo tanto, en este artículo proponemos y validamos empíricamente medidas para evaluar la mantenibilidad de procesos ETL a nivel conceptual.

3. Medición de la calidad de procesos ETL

Evaluar la complejidad estructural y conocer de forma objetiva que medidas pueden servir como indicadores de la mantenibilidad de los modelos de procesos ETL en almacenes de datos, es el marco de trabajo de este artículo. Por su parte, las propiedades estructurales de un modelo, como la complejidad estructural, influyen sobre la complejidad cognitiva [5], entendible como la carga mental que se produce en las personas que tienen que tratar con el modelo. Una elevada complejidad cognitiva puede generar diferentes problemas tales como baja mantenibilidad y alta propensión a errores [5].

Considerando la importancia de los procesos ETL mencionada anteriormente y en que un mal diseño de éstos puede suponer graves errores en la demora de la carga en el AD, nuestro principal interés es evaluar la mantenibilidad de los modelos conceptuales de procesos ETL representados con diagramas de actividad UML, mediante medidas, nuestra aproximación adapta la propuesta de FMESP al modelado de procesos ETL presentado en [13], obteniendo con ello dos niveles de abstracción para la definición de

las medidas: a nivel del modelado de procesos ETL y a nivel de actividad de procesos ETL. En este caso, las medidas propuestas son a nivel de proceso y serán validadas de manera empírica.

3.1. Medidas de procesos ETL modelados a nivel conceptual

Las medidas a nivel de modelo de procesos ETL han sido definidas con el objetivo de evaluar la complejidad estructural de procesos ETL a nivel conceptual. Para ello, el método de medición utilizado ha sido el conteo del número de elementos más significativos del modelo de procesos ETL (actividades, acciones, etc.) y el número de relaciones significativas entre los elementos (flujos de entrada y/o salida).

Con el objetivo de evaluar la complejidad estructural y conocer de mane-ra objetiva cual es la mantenibilidad de los modelos de procesos ETL, se han definido un conjunto de medidas para dichos modelos. Los conceptos utilizados para la definición de las medidas se basan en la *Ontología de la Medición del Software* definida por García et al. [8]. El conjunto de medidas definidas ha sido agrupado en dos categorías: *Medidas base*, que consisten principalmente en contar elementos específicos del modelo de procesos ETL y de las cuales se ha definido un total de 11 medidas base en función de los elementos que componen el modelado de procesos ETL propuesto en [13]. *Medidas derivadas*, definidas a partir de las medidas base, permiten conocer el ratio entre dos medidas. Este grupo está compuesto por 4 medidas.

Han sido definidas 15 medidas para el modelo de procesos ETL. Las medidas y definiciones se muestran en la Tabla 1, las primeras 11 medidas son medidas base y las 4 restantes son medidas derivadas.

4. Proceso experimental

El proceso experimental propuesto por Wohlin et al. [30], que se ha utilizado para el desarrollo de los experimentos describe las siguientes etapas:

- *Definición de los objetivos del experimento*, el porqué del mismo.
- *Planificación del experimento*, que tiene como salida el diseño del experimento, y en donde se determinará el tipo de análisis estadístico que se deberá realizar.
- *Operación*, es la fase de ejecución del experimento, normalmente se lleva a cabo un entrenamiento piloto previo y luego el experimento propiamente dicho.
- *Análisis e interpretación de los resultados*, el tipo de análisis estará condicionado por la hipótesis del experimento, por el tipo de variables que se hayan elegido y el propio diseño del experimento.

Siguiendo las pautas del proceso experimental de Wohlin et al. [30] en las secciones siguientes se describen las etapas que se han llevado a cabo para validar empíricamente las medidas propuestas para modelos de procesos ETL en ADs.

Tabla 1. Medidas para modelos de procesos ETL a nivel conceptual

Medida	Definición
NAP	Número de actividades en un proceso ETL
NEE	Número elementos de entradas en el modelo de procesos ETL
NES	Número elementos de salidas en el modelo de procesos ETL
NFES	Número de flujos entrantes y salientes en el modelo de procesos ETL
NET	Número de eventos de tiempo en el modelo de procesos ETL
NEM	Número de elementos Merge en el modelo de procesos ETL
NEF	Número de elementos Fork en el modelo de procesos ETL
NEJ	Número de elementos Join en el modelo de procesos ETL
NOSEF	Número de objetos de salida de un elemento Fork en el modelo de procesos ETL
NOEEJ	Número de objetos de entrada de un elemento Join en el modelo de procesos ETL
NODS	Número de objetos DataStore en el modelo de procesos ETL
NTEES	Número total de elementos de entrada y salida en el modelo de procesos ETL $NTEES = NEE + NES$
RFESA	Ratio de flujos entrantes y salientes entre cada actividad en el modelo de procesos ETL $RFESA = \frac{NFES}{NAP}$
RDInEE	Ratio de dependencias de elementos de entrada en el modelo de proceso ETL $RDInEE = \frac{NEE}{NTEES}$
RDOutES	Ratio de dependencias de elementos de salida en el modelo de proceso ETL $RDOutES = \frac{NES}{NTEES}$

4.1. Definición

En esta etapa se determina el fundamento del experimento. Se parte de una idea experimental y se obtiene como resultado la definición del experimento. Para la definición del objetivo de la familia de experimentos se utilizó la plantilla GQM (*Goal Question Metric*) [2]:

- *Analizar* medidas relacionadas con la complejidad estructural de modelos de procesos ETL en almacenes de datos
- *Con el propósito de* evaluarlas
- *Con respecto* a su capacidad de ser usadas como indicadores de la mantenibilidad de dichos modelos,
- *Desde el punto de vista* de los diseñadores de procesos ETL para almacenes de datos
- *En el contexto de* estudiantes de licenciatura en desarrollo de software (primer experimento) y estudiantes del máster en ingeniería de software (segundo experimento), profesionales de la empresa IISA, S.A. (tercer experimento y profesisonales de la empresa TEINSA, S.A (cuarto experimento).

4.2. Planificación

En la planificación se determina cómo se ejecutará el experimento. A conti-nuación se describe cada paso llevado a cabo:

- *Seleccion del contexto*. El grupo de sujetos del primer experimento estuvo conformado por 31 estudiantes de la carrera de Licenciatura en Desarrollo de Software que se imparte en la Facultad de Ingeniería de Sistemas Computacionales de la Universidad Tecnológica de Panamá. El grupo de sujetos del segundo experimento estuvo conformado por 25 estudiantes del Máster en Ingeniería de Software de la Facultad de Ingeniería de Sistemas Computacionales de la Universidad Tecnológica de Panamá. El grupo de sujetos del tercer experimento estuvo conformado por 18 ingenieros de sistemas de la empresa Ingeniería Informática, S.A. El cuarto experimento fue desarrollado en la empresa Tecnología Inteligente, S.A. Los sujetos para el desarrollo de este experimento fueron 20 ingenieros de sistemas. Por su parte, los experi-mentos son específicos, al estar centrados en un conjunto de medidas de complejidad estructural de los modelos de procesos ETL en almacenes de datos.
- *Formulación de la hipótesis*. Este paso consiste en transformar la idea concebida para el experimento en una frase formal, es decir en hipótesis. Para formular las hipótesis se debe tener en cuenta que debemos definir la hipótesis nula (H_{0e}, H_{0m}) y la hipótesis alternativa (H_{1e}, H_{1m}). Las hipótesis plantea-das de acuerdo a las subcaracterísticas de entendibilidad y modificabilidad son las siguientes:

  Hipótesis de entendibilidad:
  - H_{0e}: No hay una correlación significativa entre las medidas de complejidad estructural y el tiempo de entendibilidad.
  - H_{1e}: Hay una correlación significativa entre las medidas de complejidad estructural y el tiempo de entendibilidad.

  Hipótesis de modificabilidad
  - H_{0m}: No hay una correlación significativa entre las medidas de complejidad estructural y el tiempo de modificabilidad.
  - H_{1m}: Hay una correlación significativa entre las medidas de complejidad estructural y el tiempo de modificabilidad.
- *Selección de las variables*. Para los cuatro experimentos la variable independiente es la complejidad estructural de los modelos conceptuales de procesos ETL. En el caso de las variables dependientes para los cuatro experimentos son dos: la subcaracterística de la usabilidad: *entendibilidad* y la subcaracterística de la mantenibilidad: *modificabilidad* de los procesos ETL. Por su parte, las variables dependientes fueron medidas a través de los tiempos de respuesta empleados por los sujetos para llevar a cabo las tareas requeridas.
- *Selección de los sujetos*. Para la selección de los sujetos de los experimentos se uso la técnica de muestreo por conveniencia (no probabilística). Los sujetos tenían amplios conocimientos de modelado (UML, bases de datos, etc.), pero no tenían conocimientos previos acerca del modelado conceptual de procesos ETL con diagramas de actividad UML. Sin embargo, a todos se les impartió una sesión de entrenamiento, sin que con ello fueran conscientes de los aspectos que se pretendían evaluar. Cada sujeto recibió 10 modelos de procesos ETL, para cada modelo se elaboraron dos cuestionarios: uno relativo a la entendibilidad del modelo, en el cual se pedía responder (Si o No) a cinco interrogantes y otro relativo a la modificabilidad en el que se propuso una serie de cuatro modificaciones a realizar.
- *Diseño experimental*. Para obtener conclusiones significativas de los experimentos se aplicarán métodos de análisis estadísticos, para la interpretación de los resultados. Se opta por un diseño intra-sujetos, ya que éste nos permite que todos los

sujetos realicen las mismas tareas experimentales. La comparación del rendimiento de las diferentes condiciones permite estudiar el efecto de la variable independiente. Esto tiene la ventaja de garantizar el control de todas las variables debido a diferencias entre sujetos [3].

El diseño experimental utilizado fue el mismo en los 4 experimentos, ya que con el fin de corroborar los resultados obtenidos en el primer experimento se llevo a cabo una réplica del mismo (segundo experimento), y de igual forma el cuarto experimento es una réplica del tercero. La variante del tercer y cuarto experimento con respecto a los dos primeros, consiste en algunos cambios en los modelos de procesos ETL del material experimental (p.e. adición de actividades), con la intención de confirmar si las medidas no validadas en los dos primeros experimentos podrían ser útiles para evaluar la usabilidad y mantenibilidad de los modelos de procesos ETL.

- *Instrumentación* Esta fase de planificación tiene que ver con los instrumentos que son necesarios para realizar el experimento y monitorearlo. Incluye los objetos experimentales, materiales varios, guías y formularios de acondicionamiento.
  - *Objetos experimentales*: han sido 10 modelos de procesos ETL desarrolla-dos a partir de estándares existentes en la bibliografía y considerando, que para que la representación sea cognitivamente eficaz, la notación del diagrama debe proporcionar mecanismos explícitos para apoyar [11]:
    - Integración conceptual: permite al lector integrar la información de diagramas separados en una representación metal coherente del pro-blema.
    - Integración perceptual: el suministro de señales de percepción (orien-tación, contextual e información direccional) ayuda a la navegación entre los diagramas.
  - *Guías de acondicionamiento*: A efectos de que los sujetos del experimento adquieran cierta destreza en el manejo del experimento se ha organizado una sesión de introducción del modelado de procesos ETL. El material también incluía un ejemplo resuelto, en el cual se indicaba la forma en que debían desarrollarse los ejercicios.

4.3. Operación

Esta es la etapa en la que se ejecuta el experimento, es decir, donde se realiza la recolección de los datos que serán analizados. La realización del experimento es un proceso que tiene como entrada el diseño del experimento y como salida principal los datos validados. La operación consta de tres pasos que se describen a continuación.

- *Preparación*. Se impartió una sesión de preparación a los sujetos antes de que empezara el experimento. No obstante, los sujetos no sabían que aspectos pretendíamos medir, ni cuáles fueron las hipótesis que se formularon. El material que se proporcionó a los sujetos fue un conjunto de diez modelos de procesos ETL y un ejemplo resuelto. Estos modelos se referían a diferentes universos del discurso, pero que eran lo suficientemente generales como para ser fácilmente entendidos por los sujetos.

- *Ejecución*. El experimento se llevó a cabo en todas y cada una de las etapas planificadas en tiempo y forma. Se repartió todo el material descrito a los sujetos y se hizo una explicación del modelado de procesos ETL. Los sujetos disponían de una hora para resolver los ejercicios (previamente se había llevado a cabo un experimento piloto para determinar un tiempo aproximado para resolver los ejercicios resultando un tiempo de cuarenta minutos).
- *Validación de los datos*. Es importante considerar cuán válidos son los resultados del experimento. Para ello, una vez recogidos los datos, controlamos si las hojas de respuestas estaban completas y si las respuestas y las modificaciones realizadas eran entendibles. Todos los cuestionarios fueron correctos. Posteriormente, se procedió a la digitalización de los datos.

4.4. Análisis e interpretación de los resultados

El conjunto de datos recolectados sin procesar se encuentran alojados en el sitio web: $http://www.lucentia.es/index.php/ETL_Process_Modelling$. Para su análisis se empleo el software SPSS versión 15, que es una herramienta estadística adecuada a efectos de potenciar el tratamiento de los experimentos. A partir del resumen de los datos se realizó por tanto su análisis estadístico. Éste resumen estaba compuesto por los valores de las medidas para cada uno de los modelos y por los promedios en los tiempos de entendibilidad y de modificabilidad. En primera instancia, para corroborar si la distribución de los datos obtenidos era normal, en cada experimento se aplicó el test de *Kolmogorov-Smirnov*. La prueba de *Kolmogorov-Smirnov* (también prueba K-S) se basa en la idea de comparar la función de distribución acumulada de los datos observados con la de una distribución normal, midiendo la máxima distancia entre ambas curvas. Como resultado de ello, se obtuvo que la distribución era no normal en todos los experimentos, por lo que se decidió utilizar un test estadístico no paramétrico como el coeficiente de correlación de *Spearman*. El coeficiente de correlación de *Spearman* es una versión no paramétrica del coeficiente de correlación de *Pearson*, que se basa en los rangos de los datos en lugar de hacerlo en los valo-res reales [6]. Resulta apropiada para datos ordinales, o los de intervalo que no satisfagan el supuesto de normalidad. Para nuestros experimentos se utilizó un nivel de significancia 0.05 (α = 0.05) lo cual indica la probabilidad de rechazar la hipótesis nula cuando es cierta, es decir, el nivel de confianza es del 95 %. Para una muestra de tamaño 10 y un α =0.05 el umbral de *Spearman* para aceptar H_{0e} y H_{0m} es 0,6320. Por su parte, cada una de las medidas fue correlacionada separadamente con los tiempos de entendibilidad y modificabilidad.

En los resultados del análisis de correlación de los cuatro experimentos para los tiempos de *entendibilidad* y de *modificabilidad*, se obtuvo que las medidas que tienen correlación con los tiempos de respuesta para las tareas relativas a la subcaracterística de *entendibilidad* y que fueron validadas en al menos dos de los cuatro experimentos fueron: NES (Número de elementos de salidas en el modelo de procesos ETL), NFES (Número de flujos entrantes y salientes en el modelo de procesos ETL), NEM (Número de elementos Merge en el modelo de procesos ETL) y NTEES (Número total de elementos de entrada y salida en el modelo de procesos ETL) (ver Tabla 2).

Respecto a la subcaracterística de *modificabilidad* se obtuvo que las medidas que tienen correlación con los tiempos de respuesta para las tareas relativas a la modificabi-

Tabla 2. Resultados de la correlación de Spearman para los tiempos de entendibilidad

	Tiempos de Entendibilidad			
Medida	Experimento No.1	Experimento No.2	Experimento No.3	Experimento No.4
NAP	- 0.022 p=0.952	- 0.022 p=0.952	- 0.190 p=0.598	0.217 p=0.547
NEE	0.205 p=0.569	0.278 p=0.439	0.103 p=0.776	-0.270 p=0.450
NES	0.637(*) p=0.048	0.683(*) p=0.029	0.657(*) p=0.039	0.734(*) p=0.016
NFES	0.618(*) p=0.057	0.461 p=0.180	0.687(*) p=0.028	0.693(*) p=0.026
NET	0.145 p=0.690	0.442 p=0.200	0.315 p=0.376	0.262 p=0.464
NEM	0.728(*) p=0.041	0.397 p=0.331	0.637(*) p=0.048	0.893(*) p=0.003
NEF	- 0.531 p=0.175	-0.424 p=0.296	0.820(*) p=0.013	-0.424 p=0.396
NEJ	-0.018 p=0.934	- 0.295 p=0.570	-0.164 p=0.756	0.237 p=0.651
NOSEF	-0.257 p=0.540	- 0.323 p=0.436	0.294 p=0.479	-0.101 p=0.811
NOEEJ	0.224 p=0.562	-0.214 p=0.581	0.560 p=0.117	-0.455 p=0.219
NODS	0.495 p=0.146	0.760(*) p=0.011	0.414 p=0.234	0.322 p=0.363
NTEES	0.718 (*) p=0.019	0.793(*) p=0.006	0.749(*) p=0.013	0.756(*) p=0.088
RFESA	0.508 p=0.134	0.420 p=0.227	0.355 p=0.314	0.355 p=0.314
RDInEE	-0.070 p=0.848	-0.014 p=0.969	-0.324 p=0.361	-0.324 p=0.361
RDOutES	0.215 p=0.550	0.152 p=0.675	0.460 p=0.181	0.460 p=0.181

lidad y que fueron validadas en al menos dos de los cuatro experimentos fueron: NES (Número de elementos de salida en el modelo de procesos ETL), NEM (Número de elementos Merge en el modelo de procesos ETL) y NTEES (Número total de elementos de entrada y salida en el modelo de procesos ETL), (ver Tabla 3).

5. Conclusiones y trabajo futuro

En este trabajo se han presentado los resultados obtenidos a partir de la realización de expe-rimentos, los cuales se han llevado a cabo con el objetivo de analizar y evaluar la complejidad estructural de los modelos de procesos ETL a nivel conceptual. Estas medidas fueron validadas empíricamente. Los experimentos se han centrado en el estudio de la relación entre las medidas propuestas y la entendibilidad y la modificabilidad de los modelos de procesos ETL. Para lo cual, se han considerado los tiempos que invirtieron los sujetos en realizar los ejercicios relacionados con la subcaracterística de la usabilidad: *entendibilidad* y la subcaracterística de la mantenibilidad: *modificabilidad*.

Como resultado de este estudio, podemos concluir que las medidas de NES, NFES, NEM y NTEES son buenos indicadores de mantenimiento. Estas medidas proveen información sobre la mantenibilidad de los modelos de procesos ETL. A mayor mantenibilidad de los modelos de procesos ETL se pueden lograr beneficios significativos en el desarrollo de los almacenes de datos en los siguientes aspectos: i) Más facilidad para resolver los cambios en los modelos, ii) Reducción en los costos y en los esfuerzos necesarios para realizar los cambios en los modelos.

Como trabajo futuro inmediato en el contexto de la familia de experimentos se tiene previsto realizar un nuevo diseño experimental, con el fin de confirmar si las medidas no

Tabla 3. Resultados de la correlación Spearman para los tiempos de modificabilidad

	Tiempos de Modificabilidad			
Medida	Experimento No.1	Experimento No.2	Experimento No.3	Experimento No.4
NAP	0.155 p=0.670	0.155 p=0.670	0.185 p=0.815	-0.113 p=0.775
NEE	0.277 p=0.438	-0.195 p=0.989	-0.175 p=0.629	0.179 p=0.620
NES	0.705(*) p=0.023	0.650(*) p=0.042	0.487 p=0.153	0.690(*) p=0.027
NFES	0.634(*) p=0.049	0.575 p=0.082	0.037 p=0.919	0.010 p=0.979
NET	0.034 p=0.927	0.138 p=0.704	0.454 p=0.188	0.662(*) p=0.037
NEM	0.812 (*) p=0.014	0.714(*) p=0.047	0.645(*) p=0.084	0.464 p=0.247
NEF	- 0.659 p=0.016	-0.558 p=0.151	0.618 (*) p=0.057	0.065 p=0.903
NEJ	0.143 p=0.787	0.065 p=0.903	-0.272 p=0.720	-0.272 p=0.720
NOSEF	0.353 p=0.391	- 0.225 p=0.593	0.354 p=0.323	-0.204 p=0.628
NOEEJ	0.080 p=0.838	-0.165 p=0.672	0.725(*) p=0.27	-0.478 p=0.193
NODS	0.457 p=0.189	0.408 p=0.241	0.244 p=0.496	0.531 p=0.114
NTEES	0.776(*) p=0.008	0.689(*) p=0.028	0.458 p=0.183	0.743(*) p=0.014
RFESA	0.452 p=0.190	0.328 p=0.354	0.043 p=0.905	0.146 p=0.687
RDInEE	-0.249 p=0.488	-0.277 p=0.439	-0.333 p=0.348	-0.173 p=0.634
RDOutES	0.259 p=0.471	0.232 p=0.529	0.441 p=0.207	0.367 p=0.296

validadas en esta familia de experimentos pueden ser útiles para evaluar la usabilidad y mantenibilidad de los modelos de procesos ETL en almacenes de datos, o pueden ser candidatas a ser descartadas. Además, se pretende llevar a cabo casos de estudio usando modelos de procesos ETL reales de empresas.

Agradecimientos

Este trabajo es soportado por los proyectos ESPIA (TIN2007-67078) del Ministerio de Educación y Ciencia de España y QUASIMODO (PAC08-0157-0668) de la Consejería de Educación y Ciencia de Castilla-La Mancha, España. Lilia Muñoz dispone de una beca de la Secretaria Nacional de Ciencia, Tecnología e Innovación y el Instituto para la Formación y Aprovechamiento de Recursos Humanos, de la República de Panamá, Jesús Pardillo y Jose-Norberto Mazón disponen de becas AP2006-00332 y AP2005-1360 respectivamente, del Ministerio de Educación y Ciencia de España.

Referencias

1. Balta, M., Felea, V.: Using Shannon Entropy in ETL Processes. IEEE Computer Society. ISBN:0-7695-3078-8 pp. 151-156, sep.(2007)
2. Basili V., and Rombach, H.: The TAME Project: Towards Improvement-Oriented Software Environments. IEEE Transactions on Software Engineering, pp. 758-773. (1988)
3. Basili, V., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Transactions on Software Engineering. 25(4) pp. 435-437.(1999)

4. Berenguer, G., Romero, R., Trujillo, J., Serrano, M., Piattini, M.: A Set of Quality Indicators and Their Corresponding Metrics for Conceptual Models of Data Warehouses. In: DaWaK 2005: 95-104
5. Briand, L., Wust, J., Lounis, H.: A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study. 21st In't Conf. Software Engineering, Los Angeles, 345-354 (1999)
6. Corder, G., D. Foreman. Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach", Wiley (2009)
7. García, F., Piattini, M., Ruiz, F., Canfora, G., Visaggio, C.: FMESP: Framework for the modeling and evaluation of software processes. Journal of Systems Architecture 52(11): 627-639 (2006)
8. Garcia, F., Bertoa, M. F., Calero, C., et al., Towards a Consistent Terminology for Software Measurement. Information and Software Technology, 48(8) 631-644 (2006)
9. Inmon, W.: Building the Data Warehouse. QED Press/John Wiley, (1992)
10. Jarke, M., Lenzerini, M., Vassiliou, Y., and Vassiliadis, P.: Fundamentals of Data Warehouses. second edition ed: Springer-Verlag., (2002)
11. Kim, J., Hahn, J., Hahn, H.: How Do We understand a System with (So) Many Systems Research Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning. Information Systems Research 11(3). pp. 284-303. (2000)
12. Mazón, J-N., and Trujillo, J.: An MDA approach for the development of data warehouses. Decision Support Systems, 45(1): p. 41-58. (2008)
13. Muñoz, L., Mazón, J-N., Pardillo, J., Trujillo, J.: Modelling ETL Processes of Data Warehouses with UML Activity Diagrams pp. 44-53 LNCS 5333, Monterrey, (Mexico), November 9-14 (2008)
14. OMG,Software Process Engineering Metamodel Specification, adopted specification, version 1.0. Object Management Group, Inc., April, (2008)
15. OMG: MDA Guide (draft version 2). http://www.omg.org/docs/omg/03-06-01.pdf(2003)
16. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y.: Design Metrics for Data Warehouse Evolution. In: ER 2008: 440-454
17. Pardillo, J., Mazón, J-N., Trujillo, J.: Model-Driven Metadata for OLAP Cubes from the Conceptual Modelling of Data Warehouses. In: DaWaK 2008: 13-22
18. Prat, N., and Cherfi, S.: Multidimensional Schemas Quality Assessment. In: CAiSE Workshops (2003)
19. Serrano, M., Calero, C., Trujillo, J. Piattini, M.: Metrics for data warehouse conceptual models understandability. Information & Software Technology 49(8): 851-870 (2007)
20. Shah, F.: Data integration strategies for reliable information delivery. DM Review Magazine. November (2005)
21. Shannon, C.: A Mathematical Theory of Communication. The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, July, October, (1948)
22. Shilakes, C., Tylman, J.: Enterprise Information Portals.Enterprise Software Team http://sagemaker.com/ company/downloads /eip/indepth.pdf
23. Simitsis, A., Vassiliadis,P.: A Methodology for the Conceptual Modeling of ETL Processes: In CAiSE Workshops. (2003)
24. Simitsis,A., Vassiliadis, P. and Sellis, T.: State Space Optimization of ETL Workflows. IEEE Trans. Knowl. Data Eng., 17(10)1404-1419, (2005)
25. Solomon, M. Ensuring A Successful Data Warehouse Initiative. Information Systems Management, 22:1, (2005) 26-36.
26. Trujillo, J. and Luján, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: ER 2003: 307-320
27. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. Proceedings of DOLAP, (2002)

28. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Modeling ETL Activities as Graphs. In: DMDW 2002: 52-61
29. Vassiliadis, P., Simitsis, A., Terrovitis, M., Skiadopoulos, S.: Blueprints and Measures for ETL Workflows. In: ER 2005: 385-400
30. Wohlin, C., Höst, R.P., Ohlson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction, K.A. Publishers, Editor. (2000).

ECAPRIS: Metodología ágil de medición de calidad y productividad en PyMEs

Astrid Duque[1], Joaquin Lasheras[2], and Ambrosio Toval[1]

[1] Grupo de Investigación de Ingeniería del Software. Departamento de Informática y Sistemas, Universidad de Murcia,Campus de Espinardo. 30071 Murcia. España
[2] CENTIC - Centro Tecnológico de las Tecnologías de la Información y las Comunicaciones. C/ Condes de Barcelona, 5 2ªPlanta 30007 Murcia
{astrid.duque@alu.um.es,joaquin.lasheras@centic.es,atoval@um.es}

Resumen Actualmente, existe un gran interés en la medición, porque permite evaluar los métodos y técnicas que se utilizan en las empresas, y ayuda en las certificaciones como CMMI o ISO. Existen diferentes planes y programas de medición, pero son de difícil implementación en las pequeñas y medianas empresas (PyMEs) debido a sus limitaciones existentes en recursos, tiempo, conocimiento en medición o presupuesto. En este artículo se presenta ECAPRIS (Evaluación de la CAlidad y la Productividad al implantar un método de Ingeniería de Software), un método de medición ágil basado en Goal Question Metric (GQM) y Capability Maturity Model Integration (CMMI) para evaluar la mejora en calidad y productividad que conlleva la implantación de nuevos métodos de Ingeniería de Software en las PyMEs. ECAPRIS se ha evaluado en el marco de un proyecto real de implantación de un método de Ingeniería de Requisitos en 5 PyMEs de la Región de Murcia.

Key words: Ingeniería de Requisitos, Pymes, Medición, Medición de calidad y productividad, Experiencia industrial.

1. Introducción

La medición de software se hace necesaria para incrementar los niveles de calidad y productividad del software, no sólo porque contribuye a conocer las mejoras que se presentarían en la empresa al implantar nuevos métodos de Ingeniería de Software, sino porque además es la fase inicial para que las empresas logren niveles de madurez y obtengan certificaciones como ISO (International Organization for Standardization) o CMMI [1]. Para hacer las mediciones, existen diferentes tipos de programas y planes de medición, ISO/IEC 15939 [2], PSM [3], GQM [4], GQ(I)M [5], sin embargo, estos métodos están orientados a las grandes empresas, sin tener en cuenta que, en la mayoría de los países, la industria de software se compone principalmente de pequeñas y medianas empresas (PyMEs). En España, en 2007, un 99,8 % de las empresas de software eran PyMEs [6], en Méjico en el 2002 un 92 %[7], en Brasil, en el 2001 el 69 % y en Alemania, en el 2000, el 77 % [8]. Los métodos y programas de medición existentes son de difícil implementación en las PyMEs, debido a las limitaciones que éstas presentan en recursos, tiempo, escasa formación, inexperiencia en medición, presupuestos,

etc [8]. Se han creado algunos marcos metodológicos para aplicación de programas de medición en PyMEs [8] y [9], éstos son de gran utilidad, sin embargo, es complejo introducir un marco metodológico completo en las PyMEs, cuando éstas no han tenido un acercamiento con la medición.

Con el fin de introducir y familiarizar a las PyMEs con los métodos de medición, se definirá ECAPRIS, un método de medición ágil para medir calidad y productividad, basado en GQM [4] y CMMI [1] y que además sigue algunos de los principios presentados en [7] para definir programas de medición para PyMEs. ECAPRIS consta de tres actividades principales y subtareas simples que pueden ser fácilmente implementadas por los stakeholders que desarrollen un proyecto en las PyMEs.

ECAPRIS supone un punto de partida para que las PyMEs se inicien en los procesos de medición y puedan aspirar a obtener certificaciones como ISO o CMMI. Esta metodología no pretender reemplazar a ninguno de los programas de medición antes descritos, sino que busca principalmente familiarizar a las PyMEs con el proceso de medición e introducir de una forma sencilla, los métodos de medición, para que éstas se vean motivadas por iniciar este proceso y puedan mas adelante implantar programas y planes de medición más completos, e integrarlos en sus procesos de desarrollo.

En este artículo se presenta el método ECAPRIS y se muestra a manera de ejemplo, un caso de estudio, en el que se implanta durante la fase de Ingeniería de Requisitos a través de la aplicación del método de reutilización de requisitos basado en estándares denominado SIREN (SImple REuse of software requiremeNts) [10]. Este caso de estudio se enmarca dentro de un proyecto financiado denominado GARTIC (Gestión Automatizada de Requisitos basada en reutilización para PyMEs del sector TIC) donde participaron 5 empresas TIC de la Región de Murcia.

El resto del artículo se estructura de la siguiente forma: la sección 2 describe el método ECAPRIS, la sección 3 presenta la aplicación del método en las 5 empresas mencionadas, la sección 4 reune los resultados y lecciones aprendidas, y, finalmente, la sección 5 muestra las principales conclusiones.

2. Método ECAPRIS

ECAPRIS es un método de medición ágil basado en GQM, cuyo principio básico es que la medición debe estar orientada a un objetivo [4], y en CMMI, concretamente en su proceso de medición y análisis, donde se incluyen medidas que soportan actividades de mejoramiento del proceso y el producto [1]. Además el método, sigue algunas de las pautas presentadas en [7] para la elaboración de planes de medición en PyMEs. En la Fig. 1 se muestra el método ECAPRIS instanciado al caso de estudio de SIREN dentro del proyecto GARTIC para facilitar su comprensión. El método consta de tres actividades principales: DR. *Definir Roles*, IP. *Identificar proyecto anterior* y SEPM. *Seguir proceso de medición*. Cada actividad consta a su vez de varias tareas que cuentan con artefactos para la captura de información (cuestionarios y documentos explicativos para los stakeholders), que pueden ser generadas por las PyMEs de acuerdo a sus necesidades de información. A continuación se explican las actividades y tareas del método, para luego en la sección 3 ver como quedan instanciadas dentro del caso de estudio.

DR. Definir Roles Para llevar a cabo la medición se definen los roles de los implicados en ella [7]. Los stakeholders que desarrollarán el proyecto serán los encargados de ejecutar el proceso de medición. Éstos completarán los datos necesarios para la medición y definirán, de acuerdo a la madurez de la empresa, cada una de las actividades del método ECAPRIS.

IP. Identificar Proyecto Anterior

IP.1. Identificar proyectos similares/históricos. Identificar proyectos anteriores realizados en la organización que sean similares (de acuerdo con un conjunto de indicadores previamente establecidos: recursos, objetivos, requisitos, etc) al que se desea realizar con el nuevo método, y en los que se tengan registros históricos.

IP.2. Identificar métricas. Identificar las métricas que han sido utilizadas en dichos proyectos. Se busca información de defectos, tiempo y demás medidas que se generaron durante la ejecución del proyecto similar.

SEPM. Seguir Proceso de Medición

SEPM.1. Definir los objetivos de medida. Los objetivos de medida documentan el propósito por el cual se están haciendo las medidas y el análisis. Tiene su base en el proceso de medición y análisis de CMMI [1] y el objetivo general de la evaluación. En nuestro caso, dicha evaluación será la mejora en la productividad y la calidad del proceso y el producto software.

SEPM.2. Definir las medidas. Con base en CMMI [1], tanto para el proyecto similar/históricos como para el nuevo proyecto, se definen las medidas cuantificables, que refinan los objetivos de medida, las cuales pueden ser: medida real del esfuerzo el cual es medido en horas/desarrollador y medidas de calidad (número de defectos por ”gravedad, dificultad, importancia”).

SEPM.3. Definir el almacenamiento de datos. Esta tarea consta de dos subtareas que se describen a continuación:

- SEPM.3.1. Identificar Obstáculos. Obstáculos que se presentan en la PyMEs [11] y recomendaciones de solución:
  - Limitación de recursos y tiempo: los recursos que se asignan a los proyectos son ajustados, por lo tanto, las tareas de medición no se hacen con el cuidado y exactitud necesarios. Las actividades del proceso de medición son llevadas a cabo por la misma persona que ejecuta el proyecto.
  - Inexperiencia en medición: debido a esto, es posible que las medidas no indiquen de forma clara los objetivos a los que apuntan. Objetivos cuantificables bien definidos y guias de apoyo.

  - Escasa formación: debido a la limitación en presupuesto, no se forma a los stakeholders en la medición. Por lo tanto, no existe una cultura de medición ni se conocen sus beneficios. Actividades del método de medición fáciles de entender por los stakeholders, guias con información de los beneficios del uso de las métricas.
  - Dudosa fiabilidad de las actividades de medición: si los procesos de recolección de datos, análisis, interpretación y comunicación de resultados son insuficientes, mal organizados o poco fiables, se crea desconfianza en el uso del programa de medición. Procesos de recolección de datos organizados y completos.
  - Costos: no se conoce la relación costo/beneficio de la implantación de un programa de medición y es por ello que las empresas no usan estos programas. Utilización de los recursos disponibles en la empresa.
- SEPM.3.2. Crear Herramientas. Se deben emplear herramientas automatizadas y entrevistas o cuestionarios con preguntas concretas, para facilitar el uso de las métricas, mejorar la precisión de los datos obtenidos y evitar la disminución de la productividad del stakeholder, debido a esta tarea [11].

SEPM.4. Obtener Datos. Se registran los datos relacionados con las medidas definidas en la subtarea SEPM.2. tanto para el proyecto similar/históricos, como para el nuevo proyecto. Se captura la información relacionada con tiempos y defectos. Para la medición de defectos se hace el estudio de los estándares de calidad para cada fase del proyecto y se registra información de tiempos y defectos:

- Esfuerzo total de creación del proyecto, según la fase: horas/hombre para creación del documento de requisitos, horas/hombre de desarrollo, etc.
- Tipos de defectos generados por fase: tipos de defectos generados en los requisitos, de acuerdo al estándar IEEE 830 [5].
- Esfuerzo invertido en corrección de errores: expresado en horas/hombre.
- Esfuerzo invertido en revisión del proyecto por fases: requisitos, implementación, implantación, etc.

SEPM.5. Analizar los Datos. Para obtener los resultados, se hace el análisis de los datos recolectados, siendo altamente recomendado automatizar este proceso. Finalmente, se interpretan los resultados de acuerdo a los objetivos de medida definidos. Se comparan los resultados del proyecto anterior y del nuevo proyecto, para conocer la mejora en la calidad y productividad al implantar el nuevo método de Ingeniería de Software.

3. Aplicación del Método ECAPRIS - Caso de Estudio

El proyecto GARTIC, realizado por el Grupo de Ingeniería del Software (GIS) de la Universidad de Murcia, el Centro Tecnológico de las Tecnologías de la Información y las Comunicaciones (CENTIC) y 5 pequeñas y medianas empresas de la región de Murcia, tiene como objetivo la introducción, en las empresas participantes, de un método de Ingeniería de Requisitos, basado en reutilización mediante técnicas, proceso, guías y herramienta CARE (Computer-Aided Requirements Engineering), en concreto el método SIREN [10].

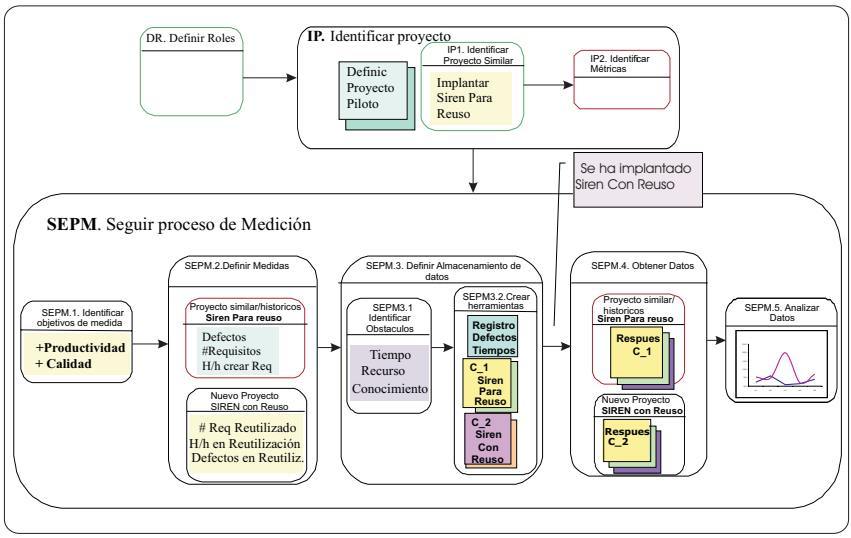


Figura 1. Método ECAPRIS y su aplicación al proyecto GARTIC.

El proyecto GARTIC se inició con un curso de formación en la metodología SIREN y la herramienta CARE RequisitePro [12] para los stakeholders de las PyMEs, y continuó con la implantación de SIREN en dos fases: en la primera fase, se hace una identificación de dominios candidatos (verticales y horizontales) de interés para las empresas y con base en los resultados y necesidades de la empresa, se implanta SIREN para reuso, obteniéndose el catálogo de requisitos para el dominio seleccionado. En la segunda fase, se implanta SIREN con reuso, en la cual se desarrolla un Proyecto Piloto con documentos de requisitos específicos, que se generan a partir de la reutilización del catálogo de requisitos creado en la fase inicial. En estas dos fases de implantación de SIREN se centra el método ECAPRIS de medición.

El objetivo del estudio es analizar el proceso de desarrollo de aplicaciones utilizando el método SIREN, con respecto al desarrollo sin su implantación, con el propósito de evaluar las variaciones en la calidad y la productividad del proceso de gestión de requisitos y del producto software, en este caso, un documento de requisitos específicos y un catálogo de requisitos. Para ello se definen dos hipótesis:

- Hipótesis 1. El desarrollo con SIREN produce software de mayor calidad que el desarrollo sin SIREN.
- Hipótesis 2. El desarrollo con SIREN mejora la productividad de los stakeholders.

Con el fin de demostrar las hipótesis anteriores se utilizó el método ECAPRIS definido en el apartado anterior. En la Fig. 1, se muestra la evaluación empírica del método ECAPRIS en las 5 PyMEs, con las herramientas diseñadas para desarrollar cada tarea y a continuación se describen las actividades realizadas.

3.1. Aplicación del Método

DR. Definir Roles. Cada una de las empresas asignó uno o dos analistas para desarrollar el proyecto, que son, junto con un encargado del GIS, quienes ejecutan las actividades del método ECAPRIS. El encargado del GIS crea las herramientas, cuestionarios y guías de medición, y los stakeholders de las empresas completan los cuestionarios y estudian las guías para ejecutarlas durante el desarrollo del proyecto.

IP. Identificar Proyecto Anterior

IP.1. Identificar proyectos similares/históricos. A través de un primer cuestionario, se detectó que ninguna de las empresas contaba con proyectos, con registros históricos, similares al que se iba a desarrollar con el método SIREN. Según la clasificación hecha por [13], las PyMEs del proyecto GARTIC se encuentran en un estado inicial cuyas características son: el proceso de IR es inadecuado, y depende de la habilidad y experiencia de los stakeholders, no se conoce el presupuesto ni el tiempo, los requisitos no son propiamente manejados y los documentos de requisitos pueden no existir o ser pobremente estructurados e inconsistentes. Como consecuencia, no era posible hacer el estudio por medio de comparación. Sin embargo, debido a que todas las empresas optaron por crear sus propios catálogos de requisitos reutilizables dentro de un dominio concreto, donde se implanta el método SIREN para reuso (se crea el catálogo de requisitos) fue posible asumir esta fase como el proyecto anterior/similar y obtener los registros durante dicha fase.

IP.2. Identificar métricas utilizadas. A través del primer cuestionario, se detectó además, que sólo una de las 5 empresas (20 %), contaba con registros de tiempos de tareas trazados a componentes de código, las demás no contaban con métricas ni con registros históricos de proyectos anteriores.

SEPM. Seguir Proceso de Medición

SEPM.1. Definir los objetivos de medida. Con base en un cuestionario con preguntas concretas y teniendo en cuenta el conocimiento básico, que las empresas adquirieron del método SIREN, se identificaron las posibles mejoras al implantar SIREN en las empresas. Seguidamente y teniendo en cuenta que los objetivos de medida están encaminados a la demostración de las hipótesis 1 y 2, se consultó a los expertos integrantes del GIS sobre las mejoras en calidad y productividad que se esperaban obtener al implantar el método SIREN y con base en ello, se definieron los objetivos de medida que son:

- Objetivos cuantitativos, se dividen en dos:
  - Productividad
    - Disminución del tiempo requerido para la especificación y creación de los requisitos del sistema.
    - Disminución en el esfuerzo de creación de los documentos de requisitos.
  - Calidad: disminución del número de defectos en el documento de requisitos, generados por errores en la especificación de requisitos.
- Objetivos Cualitativos, están relacionados con las mejoras percibidas por los stakeholders, se capturan a través de cuestionarios y algunos de ellos son:
  - Aumento en el nivel de uso de métodos estandarizados para la especificación de requisitos.
  - Incremento en la comprensión de las necesidades de los clientes.

SEPM.2. Definir las medidas. Con base en CMMI [10], se definieron las medidas cuantificables, que refinan los objetivos de medida. Estas medidas son:

- El proyecto similar/históricos - SIREN para reuso:
  - Número total de requisitos del catálogo.
  - Esfuerzo en creación de requisitos, medido en horas/desarrollador.
  - Número de defectos generados en el catálogo de requisitos.
- Para SIREN con reuso: las medidas cuantificables para reutilización son:
  - Requisitos no parametrizables reutilizados sin modificaciones. Requisitos que se han creado en el catálogo y que no requieren modificaciones para ser reutilizados en el nuevo proyecto.
  - Requisitos reutilizados con modificaciones. Requisitos que se han creado en el catálogo y que requieren modificaciones para ser reutilizados en el nuevo proyecto.
  - Esfuerzo invertido en la modificación de requisitos reutilizados.
  - Requisitos parametrizables que se reutilizan sin modificaciones. Requisitos parametrizables que se han creado en el catálogo y sólo requieren el ingreso del parámetro para ser reutilizados en el nuevo proyecto.
  - Esfuerzo invertido en selección de parámetros.
  - Requisitos parametrizables que se reutilizan con modificaciones. Requisitos parametrizables que se han creado en el catálogo y que requieren modificaciones además del ingreso del parámetro para ser reutilizados en el nuevo proyecto.

- Defectos en el documento de requisitos específicos generados por la reutilización de requisitos. (Trazabilidad, inconsistencias, etc.)
- Esfuerzo invertido en corrección de defectos generados por la reutilización.
- Esfuerzo total en parametrización y adecuación de requisitos reutilizados.

SEPM.3. Definir el almacenamiento de datos. Se divide en dos tareas,

- SEPM.3.1. Identificar Obstáculos. Los principales obstáculos que se presentaron al estudiar la definición de captura y almacenamiento de los registros estaban relacionados con las limitaciones en tiempo y recursos (solo el 40 % de las empresas terminaron la creación del catálogo y entregaron los cuestionarios solicitados en las fechas programadas). Además el poco conocimiento, debido a que los stakeholders no tenían formación ni experiencia en medición. Para las empresas la medición no implicó costos, ya que no se utilizó ninguna herramienta adicional para ello.
- SEPM.3.2. Crear Herramientas. En vista de lo anterior, y con el fin de disminuir el factor de riesgo, los datos que se solicitaron a los stakeholders de las PyMEs, se registran directamente en la herramienta CARE en que se genera el catálogo de requisitos, en nuestro caso, RequisitePro [12]. Para facilitar su entendimiento se creó una guía de registro de tiempos y defectos en la que se explicaba la forma en que se completaría la información. Se crearon además, cuestionarios tipo test con preguntas concretas y 5 opciones de respuesta, que fueron completados en un tiempo aproximado de 15 a 30 minutos.

SEPM.4. Obtener Datos. En esta tarea la empresa cuenta con un catálogo de requisitos probado, que cumple con las normas mínimas de calidad para requisitos según el estándar IEEE 830 [14]. Dicho catálogo contiene requisitos parametrizables y no parametrizables. Las empresas desarrollaron un nuevo proyecto, reutilizando los requisitos (implantación de SIREN con reuso). Para ello, seleccionaron del catálogo los requisitos adecuados para la aplicación concreta que se estaba desarrollando. Se debe tener en cuenta que cuando se reutiliza un requisito en SIREN, también se pueden seleccionar todos los requisitos relacionados con él, mediante dependencias inclusivas (relaciones de trazabilidad) [10]. Los datos necesarios para la evaluación son:

- Proyecto similar/históricos - SIREN para reuso: los registros se capturaron durante la implantación del método SIREN para reuso, en el que especifican y validan los requisitos reutilizables que se generarán para los catálogos de requisitos, partiendo de las normas o estándares y de un conocimiento del dominio por parte de los desarrolladores de las PyMEs de la empresa y con el apoyo de los integrantes del GIS. Este catálogo se crea en la herramienta RequisitePro y pasa por varias iteraciones a los integrantes del GIS para ser revisado y luego nuevamente a los desarrolladores, para ser corregido hasta que se cuente con un catálogo probado, que cumple las mínimas normas de calidad definidas en el estándar IEEE 830 [14]. Para obtener las medidas, se deben registrar tiempos y defectos asociados a la generación del producto (Catálogo de requisitos, SIREN para reuso). Las medidas de tiempos son directas, sin embargo, para las medidas de defectos se requiere estudiar el estándar IEEE 830 [14]. Se creó una guía, para el registro de tiempos y defectos, en la que

se explica la forma en que las empresas deben capturar la información de tiempos y defectos y el ingreso de estos datos en la herramienta RequisitePro.
- SIREN con reuso: a través de los cuestionarios, las empresas registran información cualitativa y además ingresan los datos de tiempos y defectos, que se presentan al reutilizar los requisitos del catálogo en el nuevo proyecto.

SEPM.5. Analizar datos Con base en los resultados de los cuestionarios se hace un análisis sobre los beneficios reales, tanto cuantitativa como cualitativamente, que presenta la implantación de SIREN en las PyMEs en que se desarrolla el proyecto GARTIC. En la fase de SIREN para reuso, los principales defectos fueron requisitos ambigüos, incorrectos e incompletos, mientras que en la fase de SIREN con Reuso solo se presentaron inconsistencias entre requisitos. Algunos de los resultados cuantitativos se presentan en la Tabla. 1 y los cualitativos se muestran en la sesión de resultados y lecciones aprendidas.

RESULTADOS	**EMPRESAS**				
	E1	**E2**	**E3**	**E4**	**E5**
SIREN para reuso					
# de Req. del catálogo	75	273	277	75	50
Esfuerzo de creación catálogo (H/h)	30	45	300	50	12
Esfuerzo por Req. (H/h)	0.40	0.16	1.08	0.67	0.24
SIREN con reuso					
Req. reutilizados sin modificación	13	150	120	10	25
Esfuerzo en inserción de Req. (H/h)	2	30	6	6	15
# de Req. reutilizados con modificación	6	20	10	3	12
Esfuerzo en modificiación (H/h)	1.5	5	3	1.5	5
# de Req. parametrizables sin modificación	4	0[1]	5	7	8
Esfuerzo de inserción de Req. parametrizables	0.5	0[1]	2	4,5	1
# de Req. nuevos	0[2]	200	245	10	30
Esfuerzo en creación (H/h)	0[3]	50	176	6	40
% del catálogo reutilizado	31 %	62 %	49 %	27 %	90 %
% de Req. reutilizados	100 %[4]	46 %	36 %	67 %	60 %
% de esfuerzo de reutilización Req.	20 %	18 %	2 %	17 %	56 %
% de Req. reutilizados sin modificaciones	74 %	41 %	33 %	57 %	44 %
Esfuerzo no invertido (H/h)	9.2	28.7	146.0	13.0	10.0

Tabla 1. Tabla de datos y resultados de las empresas. [1]**E2** no reutilizó requisitos parametrizables y no se genera esfuerzo . [2]**E1** no creó nuevos requisitos para el nuevo proyecto por lo tanto no se generó esfuerzo adicional[3] y el 100 %[4] de los requisitos fueron reutilizados.

3.2. Resultados y Lecciones Aprendidas

- El esfuerzo es menor en la fase de SIREN con reuso (Fig. 2a) y el número de requisitos especificados por unidad de tiempo es mayor (Fig. 2b), esto prueba la

hipótesis 2, la productividad es mayor en SIREN con reuso. Además se podría deducir que hay una relación inversamente proporcional entre número de requisitos reutilizados y tiempo invertido en especificación de requisitos en la fase de SIREN con reuso.

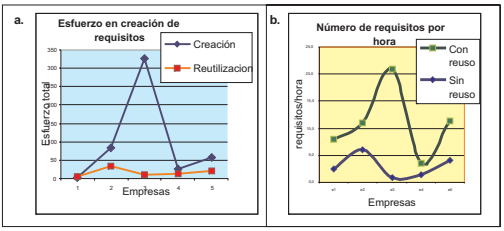


Figura 2. Gráficas con resultados de especificación de requisitos con y sin reutilización.

- El número de defectos generados en SIREN con reuso fue menor a los generados en SIREN para reuso, esto prueba la hipótesis 1 de que la calidad de los requisitos aumenta, al contarse con un catálogo de requisitos probado.
- En SIREN para reuso, la empresa dedica tiempo a la elaboración del catálogo y es por ello, que esta fase se ve disminuida la productividad de los stakeholders.
- La cantidad de requisitos reutilizados del catálogo es significativa del 36 % en adelante tabla 1.
- Algunos de los beneficios que se presentan al implantar SIREN con reuso son: disminución del número de inconsistencias entre requisitos y problemas generados por la trazabilidad; incremento en el uso de estándares de requisitos y el conocimiento del dominio de aplicación, mejoras en la documentación, trazas de requisitos, comunicación con el cliente, estimaciones de esfuerzo y gestión de requisitos; reducción de las desviaciones y agilidad en validación, al ayudar a que los requisitos se ajusten más rápidamente a las expectativas de los clientes.
- La utilización del catálogo de requisitos facilita el dimensionamiento de proyectos sucesivos, mejorando así las estimaciones de tiempo y recursos.
- Para mejorar las fiabilidad de los datos, las medidas deben registrarse en el momento en que se genera la situación, consiguiendo así más exactitud y que no haya pérdida de información, con este objetivo se solicitó a las empresas el registro de tiempos y defectos en la herramienta RequisitePro, sin embargo, las empresas no presentaron esta información en la herramienta, ni en las fechas señaladas, por lo que se hizo necesario, crear preguntas adicionales en los cuestionarios, y solicitar insistentemente los datos. Esto, confirma las afirmaciones sobre las limitaciones de las PyMEs en tiempo y recursos.
- Las herramientas para captura de información deben ser claras y apuntar siempre al objetivo deseado, con el fin de obtener resultados útiles a la hora de hacer la evaluación.
- La aplicación del método de medición inicia en el momento en el que se está planeando crear un nuevo proyecto, esto con el fin de incluir la medición en el proceso de desarrollo.

- La motivación por las métricas juega un papel importante en los buenos resultados de las medidas, es por ello que se debe hacer una concienciación de la importancia que tienen estos resultados para la empresa.
- La implantación de ECAPRIS en las empresas, ayudó a identificar algunas debilidades del método, como la necesidad de automatización del proceso de recopilación de datos de tiempos y defectos; fortalezas como la facilidad de las actividades para ser llevadas a cabo por personas que no tienen conocimiento en medición.

4. Conclusiones

- Los programas de medición son de gran importancia para las empresas que busquen mejorar sus procesos, sin embargo, la mayoría de estos programas de medición son de compleja aplicación para las PyMEs, debido a sus limitaciones en tiempo y recursos. El método ECAPRIS fue de fácil aplicación en todas las empresas, a pesar de que ninguna de ellas tenía registros históricos ni métricas anteriores, y además, los stakeholders que desarrollaban el proyecto, no tenían conocimiento ni experiencia en medición. Esto se debe principalmente a que las actividades y subtareas de ECAPRIS están definidas de forma simple para que sean entendibles por los stakeholders aunque éstos no estén familiarizados con los términos de medición.
- La medición con el método ECAPRIS no generó costos adicionales, debido a que se utilizaron herramientas como cuestionarios, guías y el software disponible en la empresa.
- La ejecución del método ECAPRIS en paralelo con la implantación de SIREN e integrada en el proceso de desarrollo, permitió capturar la información para las métricas al tiempo que se generaban los datos, logrando así, hacer la evaluación de la mejora en calidad y productividad, de una forma más ágil y confiable.
- La productividad de los stakeholders implicados en el proyecto no se vio seriamente afectada por la ejecución de las actividades y subtareas del método ECAPRIS.
- Con el método ECAPRIS, las empresas iniciaron sus conocimientos sobre métricas y mostraron interés en continuar con estas medidas y aprender más sobre ellas, con miras a iniciar sus propios procesos de certificación en CMMI.
- Tras los resultados obtenidos y las lecciones aprendidas, se han identificado trabajos futuros orientados en dos lineas:
  - Mejora del método, automatizando las actividades o identificando nuevas tareas o recomendaciones a realizar, que podrían dar lugar al refinamiento de fases o incluso a la creación de nuevas fases, como podría ser una fase de diagnóstico previa a la definición de objetivos.
  - Evaluación de la mejora: se debe comprobar que efectivamente la mejora se ha producido y seguirá produciendo (su evolución). En el caso de nuestro ejemplo, valorando la reutilización de los requisitos de los catálogos creados en nuevos proyectos que se desarrollen en la empresa.

Agradecimientos Parcialmente financiado por los proyectos DEDALO (TIN2006-15175-C05-03), MELISA:GREIS (PAC08-0142-335) y GARTIC (Contrato UMU-CENTIC 2008)

Referencias

1. CMMI, Product Team Pitsburg: CMMI for Development, Version 1.2. Technical Report CMU/SEI-2006-TR-008. (2006) http://www.sei.cmu.edu/.
2. International Organization for Standardization - ISO Geneva: ISO/IEC 15939:2002 Software engineering - Software measurement process. (2002)
3. Deptartment of Defense and US Army USA: PSM: Practical Software and Systems Measurement - A Foundation for Objective Project Management Version 4.0c. (2000)
4. R., V.S., Berghout, E.: The Goal/Question/Metric Method – A Practical Guide for Quality Improvement of Software Development. McGraw-Hill, England (1999)
5. Park, R., Goethert, W., Florac, W.: Goal-Driven Software Measurement–A Guidebook, Pitsburg. (1996)
6. Software Engineering Standards Committee of the IEEE Computer Society, IEEE-SA Standards Board Valencia, España: Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España. (Abril, 2008) http://www.inteco.es/.
7. Diaz-Ley, M., Garcia, F., Piattini, M.: Metodología para definir programas de medición en pymes: El marco mis-pyme. 13th Conference on Software Engineering and Databases, JISBD 2008 265–274 2008,Gijón, Spain.
8. Greese, C., Punter, T., Anacleto, A.: Software measurement for small and medium enterprises. 7th internation conference on Empirialc Assessment in Software Engineering (EASE), Proceedings 2003, Keele, UK.
9. Diaz-Ley, M., Garcia, F., Piattini, M.: Software measurement programs in smes, defining software indicators: A methodological framework. Product Focused Software Developmnet and Process Improvement **4589** 247–261 2007, Heildelberg.
10. Toval, A., Nicolás, J., Moros, B., García, F.: Requirements reuse for improving information systems security: A practicioner's approach. Requirements Engineering **6** (2002) 205–219 Springer.
11. Piattini, M., García, F., Garzas, J., Genero, M.: Medición y estimación del software. Técnicas y métodos para mejorar la calidad y la productividad. Alfaomega Grupo Editor , Ra-Ma, Mexico (2008)
12. Rational Software: Requisite Pro. (2000) http://www.rational.com.
13. Sommerville, I., Ransom, J.: An empirical study of industrial requirements engineering process assessment and improvement. Acm Transactions on Software Engineering and Methodology **14**(1) (2005) 85–117
14. IEEE Std 830-1998 USA: IEEE recommended practice for software requirements specifications. (1998) http://standards.ieee.org/.

A Literature Review for Obtaining PDQM v.2.0

Carmen Moraga[1], Mª Ángeles Moraga[1], Coral Calero[1], Angélica Caro[2],

[1]Alarcos Research Group – Institute of Information Technologies & Information Systems.
Dept. Information Technologies & Systems ., Univ. of Castilla-La Mancha, Spain
Carmen.Moraga@alu.uclm.es, {MariaAngeles.Moraga, Coral.Calero}@uclm.es
[2]Department of Computer Science and Information Technologies,Univ. of Bio Bio, Chile
mcaro@ubiobio.cl

Abstract. Nowadays the use of Web portals allows to obtain a vast amount of data. Those users use to carry out their daily tasks. In this context, the quality of the data is a important factor. The objective of this paper is to describe the systematic literature review (SLR) carried out in order to discover the relevant attributes in data quality for Web portals, and to evaluate the interest in data quality, in this context, since 2006 (when another SLR was carried out and a Portal Data Quality Model (PDQM) was defined). As a result, we have defined PDQM v.2.0, composed by 39 relevant attributes, for the assessment of data quality in Web portals.[1]

Keywords: Data/Information quality, Web portals

1 Introduction

The Internet is currently an important communication environment containing interesting data and useful information. However, unnecessary, out of date or erroneous data are also included. Data Quality (DQ) is therefore an actual factor in competitiveness. One means of accessing information obtained from the Internet is through a Web portal, a site which assembles information from multiple sources.

Bearing in mind the importance of DQ, the main goal of this paper is to discover the state-of-the-art in Web portal DQ through a SLR based on a previous one [1], which covered the years 1996 to 2005, and in which 33 attributes were chosen to define PDQM (Portal Data Quality Model), a model for the assessment of Web portal DQ and focused on the data consumer point of view [2]. The current SLR covers 2006 to the end of 2008, and its objective is to identify new relevant attributes.

This paper details the results and explains the process. As original contribution, we present, on the one hand, some figures about web DQ community interests in the last years and, on the other hand, the identification of new DQ attributes to update PDQM.

[1] This work is part of the projects: INCOME (PET2006-0682-01) from Ministerio de Educación and IVISCUS (PAC08-0024-5991) from the Consejería de Educación y Ciencia (JCCM) and DQNet (TIN2008-04951-E) supported by the Spanish Ministerio of Educación y Ciencia.

The rest of this paper is organized as follows. In Section 2, the SLR process, including the planning and conduction phases, is presented. The main results obtained are reported in Section 3. Section 4 explains the analysis of the obtained attributes. The interest in data quality in Web portals is described in Section 5 and conclusions and future works are outlined in Section 6.

2 Review Process

This section summarizes the activities (reader can find more information in [3]) performed in each of the two main phases of the procedure for performing an SLR, as proposed by [5]:

a) Planning the review: The most important pre-review activities are identified by the research questions(s) that the systematic review will address, and by producing a review protocol (i.e. plan) which defines the basic review procedures. In this phase, the following steps have been carried out: (1) *Identification of the need for a review;* (2) *Specifying the research questions* and (3) *Developing a review protocol, establishing* the *source selection, the search terms and the inclusion and exclusion criteria.*

b) Conducting the review: Once the protocol has been agreed, the review can begin. In this phase, the following steps have been made:

(1) *Selection of primary studies*. The search process was completed on 31/12/2008, and 4105 papers were found. Many papers were eliminated because they were duplicated, remaining 1332 papers. These papers were then analyzed and a total of 173 papers were selected, once the inclusion and exclusion criteria had been applied, 69 papers were obtained.

(2) *Data extraction and monitoring*. Once the primary studies were chosen, the following relevant information was recovered: a) Data of the paper, including the search engine, title, year, type of publication and authors; b) Data for their classification, considering the following dimensions: DQ attributes, point of view, context, application domain, quality model, measure and tool.

3 Results

This section presents the "Reporting the review" phase. The papers were classified in order to answer our research questions that are presented next.

RQ1: "Which Web data quality attributes are addressed by researchers?"

A total of 130 attributes were initially obtained. From them, 63 were defined for web or web site DQ and 67 were specific to Web portal. Bearing in mind that our objective is to select the most relevant attributes, those without description were discarded (20 of them). Next, we analyzed the attributes specific to Web Site or Web that could be applicable for Web portals selecting only 22 attributes. Finally, we analyzed the definition of the attributes, detecting that there were attributes with different names and related to the same concept, as a solution we grouped them, as is shown in Table 1., removing other 50 attributes. Therefore, a total of 39 attributes (67 for Web Portal + 22

for Web site or Web but applicable to Web portal – 50 synonymous) related to data quality attributes were eventually selected (see Table 2).

Table 1: Attributes with different name but equal concept

Attribute name	Synonymous
Accessibility	Access
Accuracy	Accurate
Amount of data	Amount of information – Appropriate Amount
Attractiveness	Detail or Appropriate level of detail or Right level of detail – Presentation
Believability	Credibility – Trustworthiness or Trustworthy– Informativeness – Reasonableness – Temporality – Veracity
Completeness	Adequacy – Sufficiency – Exactly what you need – Precision
Concise Representation	Conciseness – Association between values – Length of value
Consistent representation	Logical Consistency or Consistency
Currency	Currentness – Up-to-date or Up-to-dateness
Duplicates	Duplicity
Ease of operation	Ease of manipulation
Effectiveness	Effectively
Efficiency	Efficiently
Interactivity	Interactive
Organization	Navigability or Navigation
Readability	Human-readable – Complexity
Relevancy	Relevance
Reliability	Reliable – Free-of-error
Reputation	Authority – Authorship – Provenance
Security	Secure
Specialization	Personalization
Timeliness	Permanence – Right Time – Speed
Understandability	Ease of understanding or Ease to understand – Well-written – Comprehensiveness or Comprehensive-Coverage – Naturalness
Usability	Ease of use or Ease to use
Usefulness	Useful – Entertainment
Value-added	Quality of values or Value

Table 2: Attributes for Web portals Eventually Selected

Accesibility	Completeness	Ease of operation	Novelty	Response Time	Usefulness
Accuracy	Concise Representation	Effectiveness	Objectivity	Security	Validity
Amount of data	Consistent representation	Efficiency	Organization	Specialization	Value-added
Applicability	Currency	Expiration	Readability	Timeliness	Verifiability
Attractiveness	Customer Support	Flexibility	Relevancy	Traceability	
Availability	Documentation	Interactivity	Reliability	Understandability	
Believability	Duplicates	Interpretability	Reputation	Usability	

RQ2: "From what point of view is the Web data quality analyzed?"

The majority of the papers found are related to the data consumer point of view. This means that researchers are more concerned about the DQ with which the consumer is provided. We can therefore affirm that it is necessary to study the DQ from other perspectives, since all the perspectives are obviously related.

RQ3: "In which context the Web data quality is evaluated?"

As was previously stated, in order to attain a wide knowledge of DQ attributes the following contexts were included: Web portal, Web site and the Web in general. It must be stressed that Web sites are studied more frequently than Web portals.

RQ4: "Is a quality model defined?"

A 60% of the works defined a DQ model. This means that the works are not limited to the simple definition of attributes.

RQ5: "Do measures for Web data quality exist?"

Only 22% of the papers do not include measures unlike the situation of some years ago. It then seems that researchers have now realized the importance of measurement.

RQ6: "Does a tool with which to support the proposed approach exist?"

Although the majority of the proposals define measures, only 8 of the 69 selected papers provided a support tool, which represents 12%. This reveals the difficulty of automating the proposed measures.

4 Obtaining PDQM v.2.0

This section provides an analysis of the PDQM attributes, the attributes obtained in our SLR and how these can be used to obtain PDQM v.2.0.

4.1 Comparative between PDQM and our SLR

PDQM has 33 DQ attributes [1], selected in 2005. We have made a comparison between the PDQM attributes and the attributes detected in our SLR. Thus, a set of attributes was only selected in PDQM, first column in Table 3. The second column shows the attributes chosen from SLR and in PDQM. Finally, the last column presents 6 new attributes obtained from the SLR.

Table 3: Attributes by origin

Attributes in PDQM not obtained in our SLR	**Attributes obtained in our SLR also included in PDQM**			**New attributes obtained only in our SLR**
Customer Support	Accessibility	Consistent Representation	Reliability	Effectiveness
Documentation	Accuracy	Currency	Reputation	Efficiency
Duplicates	Amount of data	Ease of operation	Security	Readability
Expiration	Applicability	Interactivity	Specialization	Usability
Flexibility	Attractiveness	Interpretability	Timeliness	Usefulness
Response Time	Availability	Novelty	Understandability	Verifiability
Traceability	Believability	Objectivity	Validity	
	Completeness	Organization	Value-added	
	Concise Representation	Relevancy		

4.2 PDQM v.2.0

Bearing in mind that PDQM was organized into four categories: *DQ intrinsic, DQ operational, DQ contextual* and *DQ representational*,[1]. We have classified the new DQ attributes in the PDQM´s categories:

- *Verifiability* is related to the references to original sources, so it must be classified into the DQ operational category. [6]
- *Effectiveness* is related to the use of adequate analytical techniques in the design of the Web Portals [7], *Efficiency* is referred to the level of performance using appropriate amounts and types of resources [8, 9], *Usability* is focused on the ease

of use and *Usefulness* is related to the content, the use of appropriate language in a specific context, and so on. Thereby, they can be classified into the DQ contextual category [8].

- *Readability* is focused on the ease of read, so it is related to DQ representational category [10].

In Fig. 1, we show PDQM with the 6 new attributes included in boldface.

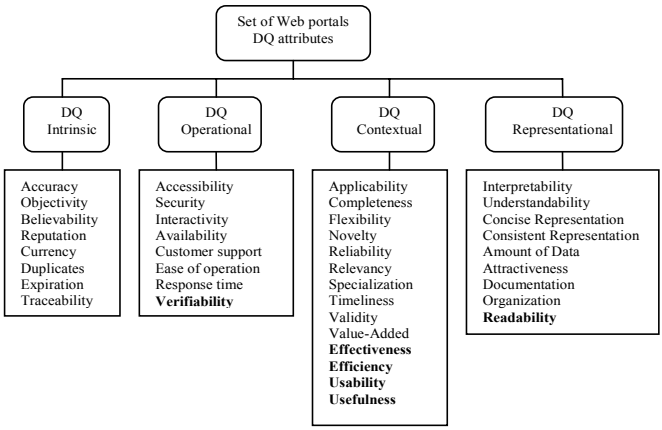


Fig. 1: PDQM v.2.0

5 Interest in data quality for Web portals

From the data obtained from the SLR, we prepared also an overview of the interest in data quality for web. For this, we have made a study of the selected papers per year. The number of papers per year that have been selected is shown in Fig. 2. This figure shows that researchers' interest in data quality in the Web is increasing.

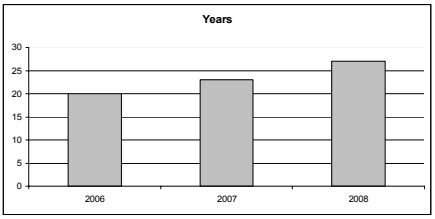


Fig. 2: Papers in each reviewed year

After analysing the types of publication of the chosen papers we can conclude that: 47 papers were published in journals, 19 in conferences, and 3 in workshops. 70% of the papers were therefore published in journals. This is perhaps due to the lack of DQ specific conferences and of topics related to DQ in non-specific conferences..

From those data we think that the interest on portals DQ is increasing.

6 Conclusions and future works

In this paper, a SLR has been carried out in order to obtain new DQ attributes to define a new version of PDQM (v.2.0) that includes 6 new attributes detected in this SLR together with the previous ones.

Moreover, realization of this SLR has led us to certain conclusions. Firstly, the majority of the papers study Web DQ from the data consumer's perspective. Secondly, it should be noted that Web sites are studied more frequently than Web portals. Thirdly, more than half the proposals define a DQ model. This means that the works are not limited to the simple definition of attributes. Fourthly, almost all the proposals define measures with which to assess the DQ level. Without measures it is obviously not possible to evaluate DQ. However, these measures are not easy to calculate automatically since only 12% of the proposals have developed a tool for their assessment.

As future work, we will study the new DQ standard of SQUARE [4], to align PDQM v.2.0 with the standard. With this, we will determine the final set of DQ attributes for PDQM v.2.0. The next step will be to validate the model and define the measures to evaluate the DQ in web portals using our model.

References

1. Caro, A., Calero, C., Caballero, I., Piattini, M.: A proposal for a set of attributes relevant for Web portal data quality. Software Quality Journal.2008; 16(4): 513-542
2. Caro, A., Calero, C., Caballero, I., Piattini, M.: Defining a data quality model for web portals. 7th International Conference on Web Information System Engineering (WISE 2006) Lecture Notes in Computer Science.2006; Vol. 4255363-374
3. Moraga, C., Moraga, M., Calero, C., Caro, A.: Towards the Discovery of Data Quality Attributes for Web Portals. In: 9th International Conference on Web Engineering. (ICWE 2009). LNCS 5648, 2009. 251-259.
4. [ISO/IEC-FDIS-25012]: Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Data quality model.2008;
5. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keely University.2007;
6. Stvilia, B., Twidale, M.B., Smith, L.C., Gasser, L.: Information quality work organization in Wikipedia. Journal of the American Society for Information Science and Technology.2008; 59(6): 983-1001
7. Yen, B., P.J-H., H., Wang, M.: Toward an analytical approach for effective Web site design: A framework for modeling, evaluation and enhancement. Electronic Commerce Research and Applications.2007; 6(2): 159-170
8. Grigoroudis, E., Litos, C., Moustakis, V.A., Politis, Y., Tsironis, L.: The assessment of user-perceived web quality: Application of a satisfaction benchmarking approach. European Journal of Operational Research.2008; 187(3): 1346-1357
9. ISO/IEC-FDIS-25012: Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Data quality model.2008;
10. De Wulf, K., Schillewaert, N., Muylle, S., Rangarajan, D.: The role of pleasure in web site success. Information & Management.2006; 43(4): 434-446

Towards a Catalogue of Patterns for defining Metrics over *i** Models

Xavier Franch, Gemma Grau

Universitat Politècnica de Catalunya (UPC)
UPC – Campus Nord, Omega building, 08034 Barcelona (Spain)
franch@lsi.upc.edu, ggrau@lsi.upc.edu

Abstract. Metrics applied at the early stages of the Information Systems development process are useful for assessing further decisions. Agent-oriented models provide descriptions of processes as a network of relationships among actors and their analysis allows discerning whether a model fulfils some required properties, or comparing models according to some criteria. In this paper, we adopt metrics to drive this analysis and we propose the use of patterns to design these metrics, with emphasis in their definition over *i** models. Patterns are organized in the form of a catalogue structured along several dimensions, and expressed using a template. The patterns and the metrics are written using OCL expressions defined over a UML conceptual data model for *i**. As a result, we promote reusability improving the metrics definition process in terms of accuracy and efficiency of the process.

Presentado en las actas de la *20th International Conference on Advanced Information Systems Engineering* (CAiSE'08), LNCS 5074, Springer-Verlag, pp. 197-212.

SMT: Software Measurement Tool

Beatriz Mora[1], Francisco Ruiz[2,3], Felix García[2]

[1]Indra Software Labs
Ronda de Toledo s/n, Ciudad Real, España
bmorar@indra.es
[2] Dep. de Tecnología y Sistemas de Información.
Escuela Superior de Informática de Ciudad Real. Universidad de Castilla-La Mancha
[3]Dep. de Matemáticas, Estadística y Computación. Universidad de Cantabria
{ francisco.ruizg;felix.garcia}@uclm.es

Abstract. SMT es una herramienta que da soporte a SMF, un marco conceptual para la medición de cualquier tipo de artefacto software o de procesos relacionados. Mediante la aplicación del paradigma de ingeniería dirigida por modelos (MDE), SMT (Software Measurement Tool) permite definir modelos de medición y de dominio que son la entrada del proceso de medición genérica del software. A partir de ellos SMT permite ejecutar mediciones, en cualquier dominio software, aplicando transformaciones a los citados modelos de medición y de dominio. SMT integra un metamodelo de medición del software (a partir del cual se definen los modelos de medición) y un lenguaje de definición de modelos de medición, SMML, que permite definir los modelos de medición de una manera gráfica, más fácil e intuitiva.

1. Introducción

La medición del software se ha convertido en un aspecto fundamental de la Ingeniería del Software. Cada vez se considera más importante y útil medir, cada vez más tipos de propiedades de más clases de entidades software. En esta línea, el marco de trabajo SMF [4], incluye todos los elementos necesarios para llevar a cabo mediciones de software, independientemente del tipo de entidad o dominio que se quiera medir. Con este marco de medición, cualquier entidad software de cualquier dominio (si se dispone del metamodelo correspondiente), puede ser medida con un metamodelo de medición de software único [2], por medio de transformaciones entre modelos. SMF tiene tres elementos fundamentales: una arquitectura conceptual, un entorno tecnológico y una metodología. EN SMF se aprovechan las ventajas del paradigma MDE y de la tecnología MDA (model-driven architecture) en el ámbito de la medición del software, en lugar de en los habituales del desarrollo o la modernización.

Una parte fundamental del proceso de medición en SMF es la definición de los modelos de medición (qué medir, cómo medirlo, cuando, porqué, quien) en base al metamodelo de medición del software [2], que incluye todos los constructores necesarios para poder añadir a los modelos la información relativa a la medición del software. Para facilitar el diseño de estos modelos se ha desarrollado “Software

Actas JISBD 2009, pp.418-421, ISBN: 978-84-692-4211-7

Measurement Modeling Language" (SMML) [3], un DSL que permite definir modelos de medición de manera visual, más fácil e intuitiva.

La herramienta SMT utiliza los conceptos y metodología de SMF implementando dos funcionalidades clave: a) la definición de los modelos de medición utilizando el DSL SMML; y b) realizar el proceso de medición genérica, obteniendo los valores resultantes, mediante transformaciones automáticas entre modelos.

2. Funcionalidad de SMT

SMT es una herramienta que da soporte al marco conceptual SMF. SMT permite realizar mediciones genéricas sobre cualquier dominio software. La herramienta está formada por la integración de dos componentes en ECLIPSE [1]. Por un lado, un editor gráfico de modelos de medición en SMML (basado en GMF) y, por otro lado, MOMENT2 para la gestión y transformación de modelos por medio de transformaciones de grafos. Las principales características y funcionalidades de la herramienta son:

- **Metamodelo de Medición del Software integrado:** a partir de este metamodelo de medición se definen los modelos de medición que son entrada del proceso de medición genérica. El metamodelo definido en ECORE contiene constructores para representar toda la información de un proceso de medición.
- **Medición de cualquier dominio:** la herramienta permite medir cualquier dominio, esto se hará mediante el metamodelo y modelo(s) de dominio correspondientes. El metamodelo deberá estar definido en ECORE.
- **Definición de modelos mediante EMF:** tanto los metamodelos de dominio, como los modelos de medición y dominio se podrán definir mediante EMF.
- **Transformaciones de grafos:** para llevar a cabo la medición del software, obteniendo los resultados correspondientes, se realizan transformaciones de grafos sobre los modelos de entrada: el de medición y el de dominio. Estas transformaciones son completamente transparentes al usuario.
- **Editor de modelos de medición:** los modelos de medición pueden definirse mediante un editor gráfico de SMML.
- **Resultados de medición:** una vez realizada la medición, los resultados se presentan en un formato de texto entendible para el usuario.

3. Arquitectura de SMT

SMT necesita gestionar y representar el conocimiento referente a la medición de cualquier entidad software de una forma integrada y consistente. Para ello, los distintos elementos de la arquitectura de SMT están organizados de acuerdo a los niveles de abstracción establecidos en el estándar MOF (Véase Fig. 1), de la siguiente manera:

- **Nivel de Metamodelo (M2):** En este nivel se incluyen: i) el metamodelo de medición, que sirve para definir los modelos de medición específicos; ii) los metamodelos de dominio, que representan a los tipos de entidades candidatas para la medición; y iii) el metamodelo de transformación de grafos, que define la especificación de cada transformación. Todos los metamodelos están definidos a partir del MetaMetamodelo ECORE y están almacenados en el *repositorio ECORE*.

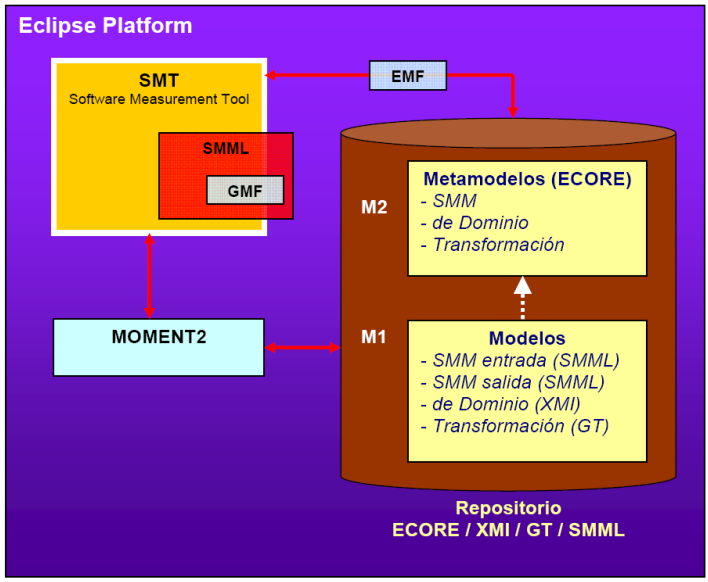


Fig. 1. Entorno SMT (Software Measuremente Tool)

- **Nivel de Modelo (M1):** En este nivel se incluyen los modelos específicos definidos mediante los metamodelos de M2: i) los modelos de medición (basados en el metamodelo de medición); ii) los modelos de dominio (definidos mediante los correspondientes metamodelos de dominio); y iii) los modelos de transformación que contienen la especificación para llevar a cabo la medición genérica. Los modelos de medición se definen mediante el editor SMML, el modelo de dominio mediante EMF y el modelo de transformación se define a partir de su metamodelo de Grafo de Transformaciones (GT).

4. Uso de SMT

Para llevar a cabo una medición del software es necesario realizar los siguientes pasos:

1. **Incorporación del metamodelo de dominio:** mediante EMF se define el metamodelo de dominio en ECORE y se incorpora al repositorio de la herramienta SMT.
2. **Creación del modelo de dominio:** a partir del metamodelo de dominio se define una instancia xmi del modelo de dominio.

3. **Creación del modelo de medición:** en base al metamodelo de medición integrado en SMF, se define el correspondiente modelo de medición mediante el editor gráfico SMML.
4. **Ejecución de la medición:** la ejecución de la medición se realiza mediante una transformación de modelos, en la que partiendo de los dos modelos de entrada (modelo de medición y modelo de dominio), se obtiene un modelo de salida que es el modelo de medición pero ampliado con los resultados de la medición. Este proceso es completamente automático y transparente al usuario (Véase Fig. 2).

Ejecutada la transformación se obtienen unos ficheros textuales con los resultados de la medición del software.

5. Conclusiones

SMT es una herramienta de medición que permite llevar a cabo mediciones genéricas para cualquier dominio software. Está basada en el paradigma MDE y la tecnología MDA. El "ingeniero de medición" define el modelo del dominio a medir y el conjunto de medidas a evaluar (en el modelo de medición) de manera gráfica y, a partir de ambos, puede obtener automáticamente los resultados de las mediciones. SMT está desarrollada en ECLIPSE.

Agradecimientos

Este trabajo ha sido parcialmente financiado por los siguientes proyectos: INGENIO (Junta de Comunidades de Castilla la Mancha, PAC08-0154-9262), ESFINGE (Ministerio de Educación y Ciencia, TIN2006-15175-C05-05) y ALTAMIRA (Junta de Comunidades de Castilla-La Mancha, Fondo Social Europeo, PII2I09-0106-2463).

Refererencias

1. *Eclipse Modeling Project Main Page*. In http://www.eclipse.org/modeling/. (2007).
2. García, F., Serrano, M., Cruz-Lemus, J., Ruiz, F. y Piattini, M.; *Managing Software Process Measurement: A Metamodel-Based Approach*, Information Sciences. 177. (2007). pp.2570–2586.
3. Mora, B., García, F., Ruiz, F. y Piattini, M.; *SMML: Software Measurement Modeling Language*, The 8th OOPSLA Workshop on Domain-Specific Modeling. Nashville (Tennessee) EU. (2008).
4. Mora, B., García, F., Ruiz, F. y Piattini, M.; *Supporting Software Process Measurement by Using Metamodels: A DSL and a Framework*, Proceedings of the Third International Conference on Software and Data Technologies ICSOFT 2008. Workshop on Metamodelling - Utilization in Software Engineering, MUSE 2008. SE/GSDCA/MUSE. pp.305-312.

Parte XII

Talleres

Integración de Aplicaciones e Información Empresarial (ZOCO, 7ª edición)

Rafael Corchuelo, David Ruiz, José Luis Álvarez, José Luis Arjona

http://www.tdg-seville.info/CfP/Zoco

La Web es el mayor repositorio de conocimiento de la Humanidad y ha conseguido un gran éxito gracias a que la mayor parte de la información que ofrece está disponible a través de mecanismos de interacción amigables y en un formato fácil de entender por las personas.

Esta es la razón por la que muchos autores hacen referencia a los sitios web como islas de datos amigables en la Web. Además, los Servicios Web y la Web Semántica están revolucionando este repositorio ya que sus tecnologías facilitan a los agentes software el acceso al mismo y abren las puertas a innumerables posibilidades de integración de aplicaciones e información desde un punto de vista semántico. El problema para utilizar estas tecnologías en sitios web ya existentes es que requieren una reingeniería que no siempre es viable (aplicaciones no desmantelables), lo que supone retos desde el punto de vista de la Ingeniería del Software.

El taller será de interés de forma específica para la comunidad de Base de Datos, ya que a día de hoy una parte importante de los retos en esta comunidad es la consulta e integración de fuentes de información en la Web.

Actas JISBD 2009, pp.425, ISBN: 978-84-692-4211-7

Apoyo a la Decisión en Ingeniería del Software (ADIS, 9ª edición)

José C. Riquelme, Roberto Ruiz, Daniel Rodríguez

http://www.cc.uah.es/drg/adis2009

En la gestión y desarrollo de sistemas de ingeniería del software se deben tomar múltiples decisiones en innumerables situaciones del desarrollo y mantenimiento de sistemas software. Las áreas en las que es preciso tomar decisiones acertadas comienzan en las evaluaciones iniciales sobre la corrección de los requisitos, evaluación de las arquitecturas, diseños, prototipos y otros elementos de las estructuras de las aplicaciones.

En las fases de diseño, codificación y pruebas los técnicos también deben tomar decisiones acerca de los productos intermedios. Además en todas las actividades de gestión nos encontramos con alternativas que el jefe de proyecto debe resolver adecuadamente. Una vez que se dispone de información cuantitativa o cualitativa, se debe proceder a su correcto uso. Estas o similares situaciones se producen en otras disciplinas, por lo que es posible reutilizar resultados ya comprobados en otras áreas. En este seminario se desea conversar tanto sobre dominios de aplicación como sobre los útiles, métodos y herramientas que ayudan a la toma de decisiones.

Así se pueden incluir técnicas como la simulación, medición, evaluación de procesos, metodologías de calidad, experimentación, modelado mediante agentes, etc. o métodos como algoritmos metaheurísticos, redes bayesianas y otros. También son de mucho interés las filosofías y enfoques de gestión (TQM, Organizational learning, Reengineering, Sistemas Complejos, etc.) en la medida que tengan repercusión sobre la toma de decisiones dentro de las múltiples actividades que se desarrollan dentro de la misma. En esta ocasión en el taller se desea examinar la aplicación de distintos aspectos de la minería de datos e inteligencia artificial a distintos aspectos de la ingeniería del software.

Actas JISBD 2009, pp.426, ISBN: 978-84-692-4211-7

Pruebas en Ingeniería del Software (PRIS, 4ª edición)

Claudio de la Riva, Peter Hodgson, Ewout van Driel, Fergus Flaherty, Juan Garbajosa, Luis Fernández Sanz, Macario Polo, Javier Tuya

http://in2test.lsi.uniovi.es/pris2009

El objetivo principal de este taller será el establecer un foro de encuentro para discusión de las actividades relacionadas con la prueba de software, tanto a nivel industrial como académico. Los principales temas de discusión responderán a las siguientes cuestiones en diferentes ámbitos: (1) Industrial: Cuál es la práctica actual en el ejercicio de la profesión y los principales problemas. Técnicas y herramientas utilizadas. (2) Académico: Cuál es el estado actual en la investigación en la prueba del software. Principales temas de investigación en los grupos nacionales e internacionales. Validación de resultados. Aportaciones a la práctica industrial. (3) Formativo: Cómo se forman los técnicos en la Industria y en la Universidad en las habilidades relacionadas con las pruebas.

Actas JISBD 2009, pp.427, ISBN: 978-84-692-4211-7

Procesos de Negocio e Ingeniería de Servicios (PNIS, 2ª edición)

Antonio Ruiz-Cortés, Manuel Resinas, Francisco Ruiz, Félix García

http://www.isa.us.es/pnis2009

El objetivo de este taller es explorar la sinergia que de manera natural se da entre procesos de negocio e ingeniería de servicios a través de las aportaciones que la ingeniería del software puede hacer al respecto. Algunas cuestiones que se abordarán son: ¿Por qué a los expertos en ingeniería del software les deben interesar los PN?; ¿Son los modelos de PN los CIM (computing independent models)? ¿Qué pasa con las dimensiones del dominio del problema versus dominio de la solución?; ¿Cómo debemos abordar la perspectiva de procesos frente a la perspectiva de sistemas, tradicional en ingeniería del software?, ¿Qué papel está llamada a jugar la tecnología BPMS (business process management systems) en la automatización de las organizaciones? ¿Cómo analizar la conformidad de los procesos de negocio con respecto a estándares, normativas, modelos de calidad? ¿Cómo afecta una evolución en un PN a la aplicación software que le está dando soporte? ¿Qué ventajas ofrece entender el proceso de desarrollo software como un proceso de negocio? ¿Cómo se puede medir la calidad de un proceso de negocio? ¿Cómo se puede utilizar una arquitectura de servicios para desplegar procesos de negocio? ¿Qué indicadores se pueden definir sobre un PN y de qué manera se puede analizar y comprobar que estos indicadores están dentro de unos valores previamente definidos?

Actas JISBD 2009, pp.428, ISBN: 978-84-692-4211-7

Desarrollo de Software Dirigido por Modelos (DSDM, 6ª edición)

José Raúl Romero, Orlando Avila-García, Vicente Pelechano

http://www.taro.ull.es/dsdm09

El objetivo principal de este taller es el de compartir experiencias a nivel académico y empresarial sobre la aplicación de técnicas y herramientas para el desarrollo de software dirigido por modelos, identificar problemas comunes, analizar las soluciones existentes y definir líneas de trabajo e investigación futuras, todo ello en el ámbito de la ingeniería del software a nivel portugués y español.

Actas JISBD 2009, pp.429, ISBN: 978-84-692-4211-7

Autonomic and SELF-adaptive Systems (WASELF, 2ª edición)

Javier Cámara, Carlos E. Cuesta, Miguel Ángel Pérez-Toledano

http://waself.lcc.uma.es/

El objetivo de este taller es reunir investigadores de áreas de ingeniería del software relacionadas con el desarrollo de self-* systems para identificar nuevos desafíos de investigación, así como para debatir y presentar modelos, técnicas, herramientas, casos industriales, y metodologías para el desarrollo de sistemas complejos capaces de adaptar dinámicamente su comportamiento. Además, el taller pretende identificar todos estos asuntos destacando la importancia de integrar logros y proponer aproximaciones genéricas a la ingeniería de self-* systems.

Actas JISBD 2009, pp.430, ISBN: 978-84-692-4211-7

Parte XIII

Tutoriales

Análisis en líneas de productos: avances, desafíos y lecciones aprendidas

David Benavides, Antonio Ruiz, Pablo Trinidad

Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla

La automatización del análisis de líneas de productos software está despertando un creciente interés entre investigadores y profesionales de la ingeniería del software. En particular, el análisis de modelos de variabilidad es uno de los temas en los que se están obteniendo más resultados y sobre el que siguen apareciendo cuestiones que motivan nuevas aproximaciones a problemas que aún siguen sin ser resueltos o cuyas soluciones pueden mejorarse.

En este tutorial se ofrecerá una panorámica de estos resultados y de algunas cuestiones abiertas, haciendo especial hincapié en los resultados incorporados y que se prevén incorporar en la herramienta FAMA .

También se darán a conocer algunas de las lecciones aprendidas durante el diseño arquitectónico de FAMA, una línea de productos en sí misma, y el soporte ofrecido a terceros para su uso e incorporación en otras herramientas.

Actas JISBD 2009, pp.433, ISBN: 978-84-692-4211-7

Herramientas Eclipse para el Desarrollo de Software Dirigido por Modelos

Cristina Vicente Chicote, Diego Alonso Cáceres

Departamento de Tecnologías de la Información y de las Comunicaciones de la Universidad Politécnica de Cartagena

En este tutorial se ofrecerá a los asistentes una revisión crítica de varias de las herramientas disponibles en Eclipse para dar soporte a todo el proceso de desarrollo de software siguiendo un enfoque de DSDM.

Entre otras, se analizarán algunas de las herramientas que proporciona Eclipse para modelado y meta-modelado, validación de modelos, construcción de editores gráficos de modelos, transformaciones modelo-a-modelo (M2M) y modelo-a-texto (M2T), etc. Para ello, se describirán, utilizando algunos ejemplos prácticos, las principales características de plug-ins tales como EMF, GMF, TOPCASED, OCL, QVT, UML2 y UML2 Tools, AMMA y ATL, openArchitectureware (oAW) y Xpand, MOSKitt, MOMENT, Kermeta, JET o MOFScript.

También se abordará la compatibilidad de estos plug-ins Eclipse con otras herramientas de modelado y otros estándares.

Actas JISBD 2009, pp.434, ISBN: 978-84-692-4211-7

Índice de autores