Gestión de Procesos en Organizaciones De Desarrollo de Software: Un Enfoque Basado en Modelos

L. García-Borgoñón[1,2], J.A. García-García[2], M. Alba[2], and F.J. Domínguez-Mayo[2]

[1] Instituto Tecnológico de Aragón, c/ María de Luna 7, 50018 Zaragoza, laurag@ita.es

[2] Universidad de Sevilla, Av. Reina Mercedes S/N, 41012 Sevilla, laura.garcia@iwt2.org, julian.garcia@iwt2.org, manuel.alba@iwt2.org, fjdominguez@us.es

Resumen La Gestión de Procesos de Negocio (Business Process Management, BPM) se centra en la definición, orquestación, documentación y mejora de los procesos de negocio en una organización. En los últimos años, siguiendo recomendaciones internacionales y guías de buenas prácticas en el desarrollo de software, las empresas de software están adoptando el BPM como mecanismo para controlar y definir cómo construir sistemas software. Sin embargo, su aplicación en este contexto de negocio no es simple debido a que el desarrollo de software está en constante evolución y a menudo incorpora novedades en los ciclos de vida, tecnologías o equipos de desarrollo, entre otros aspectos. En este trabajo se evalúa cómo un enfoque basado en modelos puede facilitar la aplicación de BPM en las organizaciones de desarrollo de software. Para ello, en primer lugar se expone una revisión de los estándares y propuestas existentes para definir procesos de software. A continuación presentamos un metamodelo de definición de procesos que se ha integrado en una herramienta mediante un perfil UML. Finalmente, se mostrará el resultado a través de NDTQ Framework, una solución actualmente en uso en varias organizaciones de software. Se concluye el trabajo con la presentación de nuevas líneas de investigación abiertas, orientadas a extender este enfoque a otros entornos.

Keywords: Proceso Software, Enfoques Basados en Modelos, Modelado de Procesos

1. Introducción

Tal y como viene siendo evidente de un tiempo a esta parte, los procesos de una organización son considerados como uno de sus principales activos, reflejo de su filosofía y esencia. Por ello, su definición e institucionalización ocupan un papel relevante en el día a día de las empresas. La Gestión de Procesos de Negocio

(Business Process Management, BPM) se centra en la definición, orquestación, documentación y mejora de los procesos en una organización.

La promesa de obtener mayor calidad unida a un incremento de la eficiencia y efectividad en el coste y esfuerzo a la hora de desarrollar sus productos, ha conllevado la adopción de procesos en múltiples dominios, cada uno de los cuales ha alcanzado la madurez en mayor o menor medida. Esto que es notable en la mayor parte de los procesos de la industria tradicional manufacturera, no se encuentra tan extendido en las organizaciones de software, en las que el uso sistemático de procesos de software y sistemas está todavía en sus etapas más tempranas.

La aparición y uso de estándares y guías de buenas prácticas ha sido habitual en los últimos años en el ámbito de las Tecnologías de la información (TI), y las organizaciones han encontrado en BPM un mecanismo con el que definir y controlar el desarrollo de sus productos. Sin embargo, la particular naturaleza de los procesos de software, con la gran implicación del factor humano, y su constante evolución, por la incorporación de nuevos ciclos de vida en el desarrollo y tecnologías, provoca que esta gestión no sea tan sencilla de realizar. Por ello, la Ingeniería de Procesos de Software se ha constituido como un área de trabajo diferenciado de los Procesos de Negocio en general, cuyo principal objetivo se centra en obtener por un lado la mejor manera de definir y diseñar los procesos de las organizaciones de software, y por otro encontrar los mecanismos más adecuados para mejorar estos procesos tanto individualmente como en su conjunto [1].

Los lenguajes de modelado de procesos software (Software Process Modeling Languages, SPML) han sido estudiados durante varias decadas [2] y generando nuevas propuestas a lo largo de este tiempo, pero todavía existen retos de futuro por alcanzar.

Uno de estos retos es la disponibilidad de mayor número de casos prácticos en organizaciones reales. Es sabido que para definir y ejecutar un proceso de software en la organizaciones es necesario: 1) un lenguaje de modelado de proceso lo suficientemente rico para la promulgación y ejecución de procesos, 2) un entorno de procesos fácil de usar que aporte flexibilidad según la categoria de proyectos y 3) un entorno de ejecución integrado [3].

En los últimos años, el uso de la ingeniería dirigida por modelos (Model-Driven Engineering, MDE) se ha establecido como un enfoque habitual en el desarrollo de software [4]. Considerando que los procesos de software son software también[5], este artículo evalúa cómo un enfoque dirigido por modelos puede facilitar la gestión de procesos en organizaciones de software, exponiendo un ejemplo práctico de su uso.

Este artículo se estructura de la siguiente forma: la Sección 2 muestra un resumen de los trabajos más destacados de lenguajes de procesos de software que se han propuesto. En la Sección 3 se expone la propuesta planteada, que consiste en un metamodelo y su implementación a través de un perfil basado en UML 2.5 [6]. La Sección 4 presenta un ejemplo práctico de su uso mediante NDTQ-Framework, una solución software que incorpora esta propuesta y que

actualmente es utilizada en varias empresas de diferente tamaño, tanto públicas como privadas. Para finalizar, la Sección 5 describe las principales conclusiones y la presentación de nuevas líneas de investigación abiertas, orientadas a extender este enfoque a otros entornos.

2. Trabajos Relacionados

Los lenguajes de modelado de procesos han sido estudiados durante varias décadas, como un mecanismo para la definición de procesos, y existe un importante conocimiento en trabajos sobre esta materia. Con anterioridad al desarrollo de la propuesta se llevó a cabo una revisión sistemática de la literatura relacionada con este ámbito, que fue presentada en [2]. A continuación se introduce un resumen de aquellos trabajos más relevantes.

La habitual comparación de procesos software con los procesos industriales de manufactura ha llevado consigo el desarrollo de muchos esfuerzos para su descripción y automatización, obteniendo como resultado los diferentes lenguajes de modelado de procesos software que se han definido a lo largo del tiempo. Uno de los aspectos que caracteriza a cada uno de estos lenguajes es la tenconlogía base que se ha utilizado para su desarrollo, algo que ha ido evolucionando con el tiempo.

Durante la época de los 90, proliferaron los lenguajes de gramáticas, entre los que se incluyen aquellos los basados en teoría de grafos, redes de Petri, reglas, e incluso lenguajes de programación. Su foco principal era la capacidad de ejecución de los procesos, pero su complejidad, formalidad y su falta de flexibilidad hicieron que no se adpotaran de forma extensiva en las organizaciones de software. De estas fechas datan algunos lenguajes tan conocidos como MARVEL[7], SPADE[8], SPELL[9] o APPL/A[10].

Otro aspecto que también tuvo un impacto negativo en su despliegue fue el hecho de que se definieran múltiples notaciones para representar los mismos conceptos de los procesos software [11], asi como la falta de consenso en cuanto a la definición de estos lo que hizo enfocar la búsqueda en un estándar como un lenguaje común para la comunidad de software.

Una posible alternativa que se planteó fue Business Process Modeling Notation (BPMN)[12] por su simplicidad y soporte a la ejecución, pero la falta de contendio semántico al tratarse de un lenguaje más orientado a procesos de negocio que a procesos de software, hizo que su uso tampoco fuera tan extendido.

Siendo Unified Modeling Language (UML) un estándar reconocido en el ámbito de la ingeniería del software, se planteó como opción a valorar, y proliferaron las propuestas desarrolladas con UML como base, en sus diferentes versiones según su año de creación. De entre ellos cabe destacar las siguientes propuestas. Di Nitto et al.[13], basada en UML 1.3, que ofrece la posibilidad de aplicar un subconjunto de UML como un lenguaje de procesos ejecutable, transformando modelos UML a modelos de flujo de trabajo ejecutables. PROMENADE [14] cubre la reutilización de procesos, entendida como la capacidad de construir nuevos procesos mediante el ensamblaje de aquellos previamente

creados, y la recolección de procesos, que consiste en la capacidad de construir procesos genéricos que puede ser reutilizados más allá de los ya existentes. El enfoque de Chou [15] utiliza un conjunto de los diagramas de actividad y de clases de UML 1.4, con un lenguaje de programación de procesos orientado a objetos, y que más adelante es enriquecido con un modelo de promulgación en un entorno de procesos software[16].

El Object Management Group (OMG) había propuesto SPEM1.1 (SOftware Process Engineering Metamodel) como estándar para la representación de procesos software, también basada en UML 1.4, pero con la publicación de UML 2.0, planteó una completa remodelación del mismo en una nueva versión, conocida como SPEM 2.0. Fue un trabajo arduo que involucró a muchas organizaciones de software, con diferentes propuestas para ser consideradas en el estándar. Una de estas propuestas fue UML4SPM [17], un metamodelo creado con cuatro objetivos claros: la expresividad, comprensibilidad, precisión y capacidad de ejecución.

Estas propuestas tiene los modelos (o los metamodelos) como tecnología base. En esta línea, pero con una definición de metamodelos diferente a la de OMG, surgió otra iniciativa denominada SMSDM [18], sobre la que está fundada la ISO/IEC 24744[19]. Utiliza el concepto de powertype para el metamodelado de metodologías de dominio.

Desde la publicación de SPEM 2.0 en 2008, numerosas propuestas que lo extienden y modifican para incorporar nuevas características han aparecido. Es el caso de xSPEM [20], eSPEM [3], MODAL [21].

Como se expone en [2], uno de los principales aspectos que se plantean en la mayoría de las secciones de trabajos futuros de las propuestas presentadas anteriormente, están relacionadas con extender su aplicación y aplicabilidad a entornos reales e intensivos dentro de la industria del software. Esto pone de manifiesto la complejidad que supone el despliegue de este tipo de lenguajes en las organizaciones cuyo ámbito de negocio es el software.

Por estos motivos, la propuesta aquí presentada pretende ser innovadora respecto al enfoque utilizado hasta ahora en el que se aboga por presentar propuestas de lenguajes con mucha semántica. No se pretende desarrollar un lenguaje para modelar procesos que incorpore numerosos conceptos, sino algo sencillo que sea fácilmente trasladable e implementable en organizaciones de software reales y, que poco a poco vayamos enriqueciendo con las lecciones aprendidas en las implantaciones. La propuesta aquí planteada se describe en las siguientes secciones y está formada por un metamodelo MOF y su implementación a través de un perfil UML.

3. Una Propuesta para Describir Procesos Software

3.1. Un Metamodelo para representar Procesos Software

Uno de los aspectos mencionado en el apartado anterior es el hecho de la falta de consenso en los conceptos relacionados con los procesos de software. Diferentes propuestas se han desarrollado para recomendar los elementos requeridos en un

proceso. Todas ellas coinciden en el principal, es decir, en el Proceso de Software, pero varian en formato, contenido y nivel de descripción. Esta fue la causa por la que el estándar ISO/IEC TR 24744[22] fue definido, para que sirviera de guía de soporte en la definición de modelos de procesos, así como para mejorar la consistencia y uniformidad en la descripción de los estándares. La propuesta que vamos a presentar a continuación utiliza los conceptos recogidos en este estándar.

La Figura 1 describe el metamodelo de esta propuesta. Los elementos que incorpora son los básicos para la definición de los procesos de una organización de software.

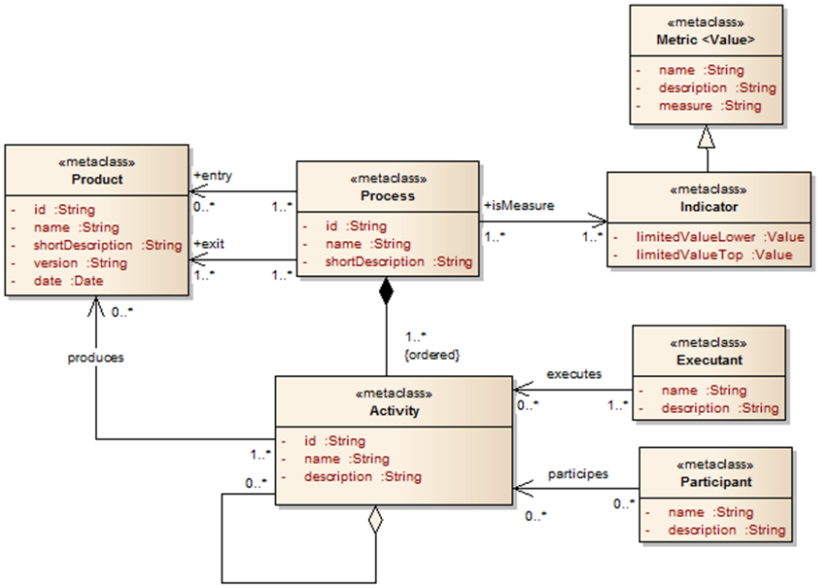

Figura 1. Propuesta de metamodelo de procesos software

La *metaclase Process* es la metaclase principal del metamodelo y representa un conjunto ordenado de acciones que son ejecutadas por alguien con el objetivo de producir algo. Para cumplir el estándar ISO/IEC TR 24744, esta metaclase tiene los siguientes atributos: *id*, el cual representa un identificador único del proceso; *name*, que proporciona el nombre del proceso; y *shortDescription* que describe cuál es el objetivo del proceso.

La *metaclase Activity* representa una acción dentro del proceso. Según recoge el estándar ISO/IEC TR 24744, una actividad puede incluir varias actividades. Esta característica se ha reflejado a través de una relación de composición recursiva sobre la *metaclase Activity*. Por otra parte, y al igual que el elemento anterior, la *metaclase Activity* dispone de los atributos *id*, que identifica de ma-

nera unívoca a la actividad; *name*, que proporciona una descripción corta de la actividad; y *description*, que proporciona una descripción más detallada.

Planteamos también dos tipos de actores que toman parte en la ejecución de una actividad, y que se clasifican en función su grado de involucración. Por un lado se contempla la *metaclase Executant*, que representa al actor que ejecuta la actividad. Por otro, la *metaclase Participant* representa el conjunto de actores que contribuyen a llevar a cabo la actividad, pero que no son directamente responsables de la misma. La diferencia entre ambos actores es que la *metaclase Executant* representa al único responsable de completar la actividad y continuar con el flujo de actividades definido en el proceso, mientras que la *metaclase Participant* hace referencia a cualquier actor que puede completar cualquier aspecto del producto que genera la actividad. En este sentido, se ha considerado necesario reflejar este aspecto debido a la heterogeneidad en cuanto a tamaño de los equipos de trabajo que existen dentro de cualquier organización dedicada al negocio del software.

Por otra parte, se ha considerado la *metaclase Product*, que representa un producto obtenido como resultado de ejecutar un proceso o como entrada al mismo. En este sentido, un producto puede ser estereotipado como un producto de entrada o como un producto de salida del proceso. Sin embargo, el producto en si mismo es generado cuando una actividad es completada. En este sentido, es necesario establecer una restricción en el metamodelo para reflejar que todos los productos de salida de un proceso deben haber sido generados por alguna de las actividades que conforman el proceso en cuestion. Esta restricción ha sido reflejada en la expresión OCL descrita en la Figura 2.

```
context Process inv:
   self.exit->forAll (p1 : Product |
            self.Activity->exist (a : Activity |
                   a.Product->exist (p2 : Product | p1 = p2)))
```

Figura 2. Restricción OCL sobre la metaclase Process

Finalmente, con el objetivo de establecer una forma para medir alguna de las propiedades de los procesos y así proporcionar mecanismos para llevar a cabo una mejora continua de los mismos, se ha planteado las metaclases *Metric* e *Indicator*. La primera pretende recoger cualquier métrica que sea necesario definir en el proceso para mejorar y medir su desempeño. Cuando este elemento tiene un valor concreto asociado hablamos de indicador, lo que permite disponer de la medida objetiva, que se considera la base para la definición de cuadros de mando.

3.2. El Perfil de Modelo de Procesos

Usan un metamodelo no es una tarea sencilla si no se establecen mecanismos ágiles para instanciarlo. En este trabajo se ha optado por definir un perfil UML

como un método para crear una sintaxis concreta de nuestro metamodelo. La elección del método para crear la sintaxis no ha sido fácil. Se ha elegido UML porque proporciona un método ágil de extensión de una manera estandarizada. Para llevar a cabo la extensión del estándar UML, éste proporciona tres mecanismos:

- **Estereotipos**. Permite definir elementos del dominio específico, es decir, los elementos del metamodelo son definidos utilizando elementos estereotipados que, una vez definidos, son los que extiendan las metaclases UML.

- **Restricciones**. Este mecanismo permite establecer condiciones sobre los elementos estereotipados del metamodelo, para describir, entre otras, las condiciones que han de verificar un modelo bien formado. Un lenguaje de restricción muy utilizado es OCL.

- **Valores etiquetados** o **Tagged Value**. Son meta-atributos adicionales que se pueden asociar a un elemento estereotipado del metamodelo.

La Figura 3 muestra parte del perfil UML 2.5 que se ha definido; no se ofrece el diagrama completo debido a sus dimensiones y porque no aporta ningún conocimiento relevante y nuevo respecto a lo ya mencionado anteriormente. En la figura se muestran algunos de los elementos estereotipados (la metaclase *Activity* y *Metric*, entre otras) junto con sus valores etiquetados (Tagged Value) y sus relaciones de extensión de las metaclases de UML 2.5 (como las metaclases *Artifact* y *Activity* de UML 2.5)

Llegados a este punto es importante comentar que el perfil UML 2.5 definido anteriormente puede ser implementado con cualquier herramienta que permita el desarrollo de perfiles, como MagicDraw, StarUML, Enterprise Architect, Rational Systems Developer o ArgoUML, entre otros. Sin embargo, la propuesta aquí presentada está basada en Enterprise Architect[1] (EA).

4. Un Ejemplo Práctico

En esta sección se aborda un ejemplo práctico de la implementación del metamodelo, para lo que se utilizarán los procesos definidos en la metodología NDT[23]. A continuación se realiza una visión general de la metodología y se utiliza uno de sus procesos para mostrar cómo el metamodelo se encuentra incluido en la herramienta.

[1] para la elección de la herramienta se llevó un estudio comparativo en colaboración con la Junta de Andalucía en el que se concluía que EA era la herramienta que mejor indicador calidad/precio arrojaba. Este estudio fue publicado y puede ser consultado desde el sitio web del grupo de investigación IWT2: http://www.iwt2.org/

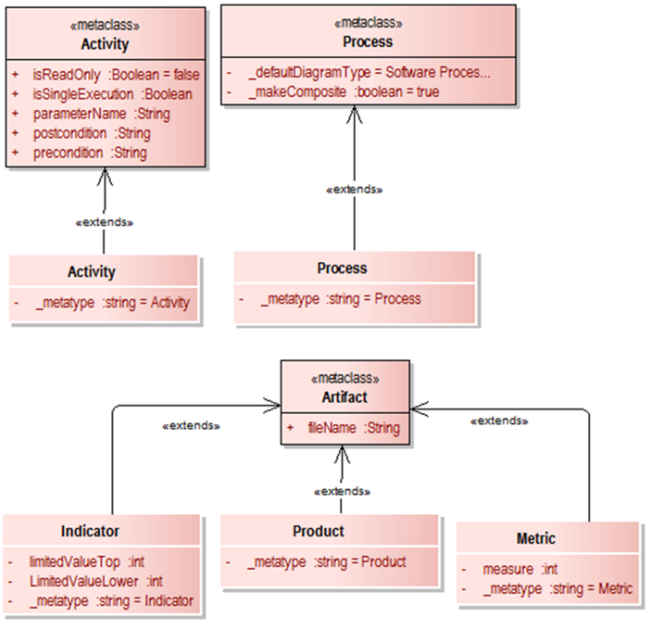

Figura 3. Perfil en Enterprise Architect para Proceso Software

4.1. Navigational Development Techniques

La metodología Navigational Development Techniques (NDT) está englobada en el paradigma MDE. Actualmente, NDT contempla y define un conjunto de metamodelos para cada una de las fases del ciclo de vida de desarrollo de software: *Estudio de Viabilidad*, *Requisitos*, *Análisis*, *Diseño*, *Implementación*, *Pruebas* y *Mantenimiento*, y establece un conjunto de reglas de transformación QVT[24] con las que es posible generar de manera sistemática unos modelos a partir de otros redundando en un menor coste asociado a la especificación del desarrollo de software.

Sin embargo, para que toda la potencia de NDT sea práctica y de utilidad en entornos empresariales es necesario disponer de herramientas que automaticen todo el proceso. Para ello, junto con NDT se proporciona un conjunto de herramientas denominado NDT-Suite [25] [26], que dan soporte a todas las fases del ciclo de vida definidas en la metodología.

Aunque, en general, la aplicación de metodologías ayuda a mejorar la calidad de los resultados obtenidos tras ejecutar un determinado proceso de una organización, es muy común que sucedan problemas que no deberían darse. Por ejemplo, uno de los más característicos en una organización de software es el hecho de que en muchos casos la elaboración de la documentación del proyecto o la aplicación de determinada fase de la metodología acaba siendo una mera formalidad, llegando a suceder en muchos casos que proyectos, inicialmente

enmarcados en una metodología, sufren cambios no registrados, produciéndose inconsistencias entre la documentación y el sistema final.

Con el objetivo de solventar este tipo de problemas, NDT ha evolucionado, proporcionando un marco de trabajo que facilite el uso de nuevos enfoques, estándares y paradigmas, y por consiguiente, nos ayude a mejorar la calidad en el desarrollo de software. Para ello, NDT utiliza los procesos de software, cada uno de los cuales está fundamentando en los principales modelos, estándares y normas relacionados en el ámbito al que se refiere. Estos procesos se engloban en seis grandes grupos:

- Grupo de procesos de ingeniería del software, basado en el ciclo de vida definido en NDT.
- Grupo de procesos de mantenimiento de software, basado en la guia de buenas practicas ITIL[27] y el modelo CMMI[28].
- Grupo de procesos de pruebas, basado en el estándar ISO/IEC 29119 [29]
- Grupo de procesos de calidad del software, basado en el estándar ISO 9001:2008[30] y en el modelo CMMI.
- Grupo de procesos de gestión de proyectos, basado en PMBOK[31] y CMMI.
- Grupo de procesos de seguridad, basado en el estándar ISO 27001[32].

4.2. El metamodelo en uso: NDTQ-Framework

Tal y como se ha referido anteriormente, para que toda la potencia de una metodología sea puesta realmente en práctica en organizaciones de software, ya sean empresariales o académicas, deberían estar soportadas por herramientas que faciliten su uso, pero también obliguen al seguimiento de la misma. Por ello, para dar soporte los procesos definidos en NDT se creó NDTQ-Framework, un marco de trabajo desarrollado en base al metamodelo presentado en la sección anterior.

NDTQ-Framework es un marco de trabajo que actualmente está siendo utilizado en diferentes proyectos reales en empresas privadas (como Airbus, entre otras) y en empresas públicas (como la Agencia de Obra Pública de la Junta de Andalucía y la Consejería de Salud y Bienestar Social de la Junta de Andalucía, entre otras) y describe todos los procesos definidos por la metodología NDT. NDTQ-Framework incluye un catálogo con un total de 26 procesos englobados en los grupos de procesos anteriormente presentados, con lo que se convierte en una herramienta estratégica para las organizaciones ya que, dependiendo de sus objetivos estratégicos, pueden optar por las implementación de aquellos procesos que más se ajusten a sus modelos de negocio.

Por otro lado, con el uso de NDTQ-Framework se sistematiza la implantación de los procesos de negocio de las organizaciones y se facilita la implantación de estándares y de buenas prácticas ya consolidadas en el mercado. Contar con el

metamodelo permite la generación automática de nuevos procesos y la facilidad del despliegue de herramientas que pueden monitorizar y medir todo el proceso mediante indicadores, lo que facilita una mejora continua de la calidad de los procesos.

Para ilustrar cómo NDTQ-Framework implementa nuestro metamodelo, vamos a utilizar como ejemplo el proceso de Ingeniería de Requisitos. En la Figura 4 se observa el mapa de actividades de este proceso.

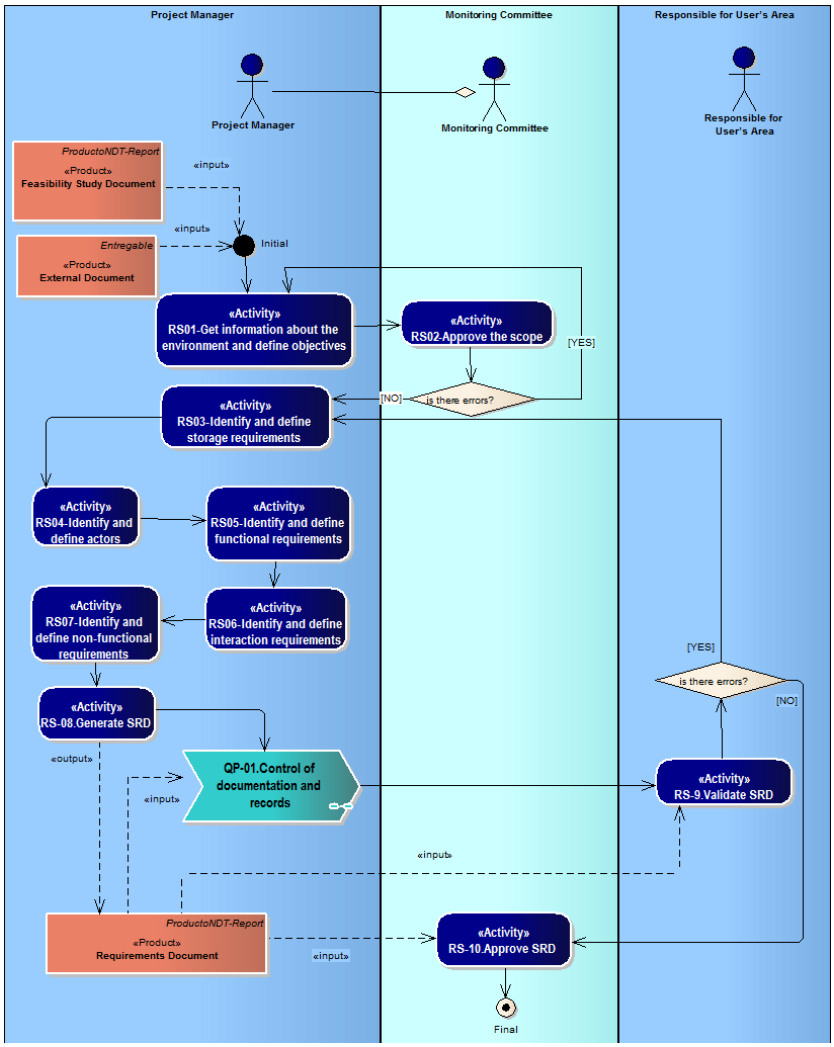

Figura 4. Proceso de Ingeniería de Requisitos

En la figura anterior se pueden identificar los elementos de nuestro metamodelo. En primer lugar, se identifica el elemento principal, el elemento *Process*, cuyo *name* es *Requirements Engineering Process*. El resto de los atributos no son visibles en esta imagen, pero son accesibles dentro del propio NDTQ-Framework.

Todos los elementos etiquetados como *RSXX-XX* constituyen las actividades del proceso, es decir, los elementos *Activity* del metamodelo. Y los dos documentos, *System Requirements Document* y *Review Report*, son *Products*.

En la imagen también queda evidente los *Executant* de las *Activities*, son cada uno de los actores que encabezan las calles en las que se encuentra la actividad. No queda reflejado, sin embargo, los *Participant* de las mismas, porque dificultaría la lectura del diagrama. De todas formas, la herramienta de forma interna lo tiene contemplado.

Por último, tanto *Metric* como *Indicator*, que no aparecen directamente reflejados en el diagrama, si que son visibles en la pestaña *Constraint* de la herramienta, como se observa en la Figura 5.

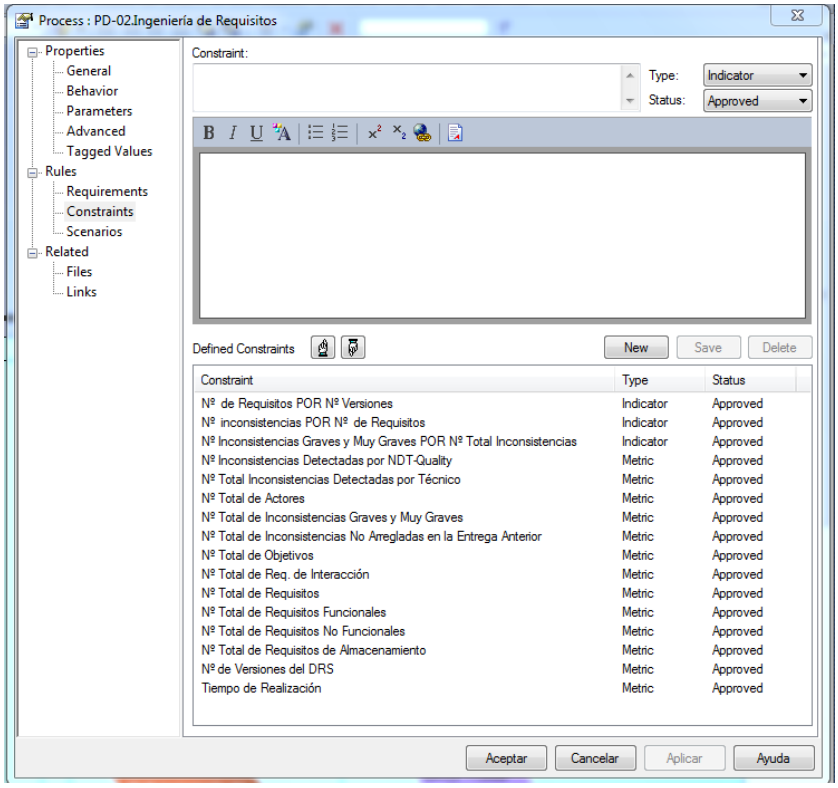

Figura 5. Pestaña Constraints de NDTQ-Framework

5. Conclusiones y Trabajos Futuros

Este artículo presenta una solución basada en modelos para definir procesos de negocio de una organización software. Después de revisar los trabajos y propuestas relacionadas observamos que uno de los principales problemas a resolver en el ámbito de los lenguajes de procesos de software era la falta de experiencias prácticas en el uso de los mismos. Por ello, para solventar dicho problema, planteamos un nuevo enfoque, sencillo y validable en organizaciones reales, que se irán enriqueciendo en trabajos futuros.

Se ha presentado un metamodelo para la definición de procesos software, que contiene todos los conceptos requeridos por la norma ISO/IEC TR 24744. También seha presentado un perfil UML 2.5 para dicho metamodelo implementado en Enterprise Architect. Haciendo uso tanto del metamodelo como del perfil, se ha desarrollado NDTQ-Framework, una solución basada en modelos para la definición de procesos. NDTQ-Framework es un marco en el que se encuentran definidos todos los procesos descritos en la metodología NDT, y se ha presentado un ejemplo práctico de cómo el metamodelo ha sido implementado en la solución.

Como trabajo futuro, estamos intentando mejorar la solución aquí presentada desde diferentes enfoques. Por un lado, estamos trabajando en extender nuestro marco con nuevos procesos, incorporando actualizaciones de modelos como ITIL o PMBOK, asi como la incorporación de otras recomendaciones. En segundo lugar, estamos trabajando en la incorporación a la solución de un mecanismo basado en modelos que nos permita la orquestación de los procesos definidos. NDTQ-Framework permite en este momento la definición, mantenimiento y documentación de los procesos, pero durante la ejecución de los mismos el mecanismo en las organizaciones es demasiado manual. Por ello, una de nuestras líneas de trabajo tiene como objetivo automatizar la ejecución y evaluación del desempeño de los procesos. Por último, se tiene como objetivo ir enriqueciendo el metamodelo con nuevos conceptos, explorando a su vez nuevos campos de uso. Por ejemplo, estamos trabajando en la extensión del mismo para dar soporte a la simulación de procesos [33], asi como en su utilización en el campo de la salud[34].

El hecho de que NDTQ-Framework esté siendo utilizado en entornos reales y en diferentes contextos nos abre una vía muy importante a la hora de validar la usabilidad de esta propuesta y ampliar el alcance de la solución.

6. Agradecimientos

Esta investigación ha sido parcialmente financiada por el proyecto NDTQ-Framework (TIC-5789) de la Junta de Andalucía, el proyecto TEMPROS (Testing Temprano y Modelos de Simulación Híbrida en la Producción de Software) del Ministerio de Educación y Ciencia (TIN2010-20057-C03-02).

Referencias

1. Acuña, S., Juristo, N.: Software Process Modeling. International Series in Software Engineering. Springer (2005)
2. García-Borgoñon, L., Barcelona, M.A., García-García, J.A., Ortega, M.A., Escalona, M.J.: Software Process Modeling Languages: a Systematic Literature Review. Information Software and Systems. Under review (2012)
3. Ellner, R., Al-Hilank, S., Drexler, J., Jung, M., Kips, D., Philippsen, M.: eSPEM–a SPEM extension for enactable behavior modeling. In: Modelling Foundations and Applications. Springer (2010) 116–131
4. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer Society **39**(2) (2006) 25
5. Osterweil, L.: Software processes are software too. In: Proceedings of the 9th International Conference on Software Engineering, IEEE Computer Society Press (1987) 2–13
6. OMG http://www.omg.org/spec/UML/: UML, Unified Modeling Language. (2012) Last accessed 01-2013.
7. Kaiser, G.E., Barghouti, N.S., Sokolsky, M.H.: Preliminary experience with process modeling in the Marvel software development environment kernel. In: System Sciences, 1990., Proceedings of the Twenty-Third Annual Hawaii International Conference on. Volume 2., IEEE (1990) 131–140
8. Bandinelli, S.C., Fuggetta, A., Ghezzi, C.: Software process model evolution in the SPADE environment. IEEE Transactions on Software Engineering **19**(12) (1993) 1128–1144
9. Conradi, R., Jaceheri, M.L., Mazzi, C., Nguyen, M.N., Aarsten, A.: Design, use and implementation of SPELL, a language for software process modeling and evolution. Springer (1992)
10. Sutton, S.M., Heimbigner, D., Osterweil, L.J.: APPL/A: a language for software process programming. ACM Transactions on Software Engineering and Methodology **4**(3) (1995) 221–286
11. Finkelstein, A., Kramer, J., Nuseibeh, B.: Software process modelling and technology. John Wiley & Sons, Inc. (1994)
12. OMG http://www.omg.org/spec/BPMN/2.0/: BPMN, Business Process Modeling Notation, Version 2.0. Last accessed 01-2013.
13. Di Nitto, E., Lavazza, L., Schiavoni, M., Tracanella, E., Trombetta, M.: Deriving executable process descriptions from UML. In: Proceedings of the 24th International Conference on Software Engineering. ICSE 2002., IEEE (2002) 155–165
14. Ribo, J.M., Franch, X.: PROMENADE: A PML Intended to Enhance Standarization, Expressiveness and Modularity in Software Process Modelling. Technical report, Citeseer (2000)
15. Chou, S.C.: A Process Modeling Language consisting of high level UML-based diagrams and low level Process Language. Journal of Object Technology **1**(4) (2002) 137–163
16. Chou, S.C.: DPEM: a decentralized software process enactment model. Information and Software Technology **46**(6) (2004) 383–395
17. Bendraou, R., Gervais, M.P., Blanc, X.: UML4SPM: An executable software process modeling language providing high-level abstractions. In: 10th IEEE International Conference on Enterprise Distributed Object Computing,EDOC'06, IEEE (2006) 297–306

18. Henderson-Sellers, B., Gonzalez-Perez, C.: A comparison of four process metamodels and the creation of a new generic standard. Information and Software Technology **47**(1) (2005) 49–65
19. ISO/IEC: ISO/IEC 24744:2007 Software Engineering – Metamodel for Development Methodologies. (2007)
20. Bendraou, R., Combemale, B., Cregut, X., Gervais, M.P.: Definition of an Executable SPEM 2.0. In: Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific, IEEE (2007) 390–397
21. Koudri, A., Champeau, J.: MODAL: a SPEM extension to improve co-design process models. In: New Modeling Concepts for Today"s Software Processes. Springer (2010) 248–259
22. ISO/IEC: ISO/IEC TR 24744:2007 Software and systems engineering Life cycle management Guidelines for process description. (2007)
23. Escalona, M.J., Aragon, G.: NDT. A Model-Driven Approach for Web Requirements. IEEE Transactions on Software Engineering **34**(3) (2008) 377–390
24. OMG http://www.omg.org/spec/QVT/1.1/: QVT, Meta Object Facility (MOF) 2.0 Query/View/Transformation. (2011)
25. García-García, J.A., Ortega, M.A., García-Borgoñon, L., Escalona, M.J.: NDT-Suite: A Model-Based Suite for the Application of NDT. In: ICWE. (2012) 469–472
26. IWT2 Disponible en http://www.iwt2.org: NDT-Suite. (2013)
27. ITIL http://www.itil-officialsite.com: Information Technology Infrastructure Library. Last accessed 01-2013.
28. Chrissis, M.B., Konrad, M., Shrum, S.: CMMi. Addison-Wesley Boston (2003)
29. ISO/IEC: ISO/IEC 29119 Software Engineering – Software Testing Standard. International Organization for Standardization
30. ISO/IEC: ISO 9001:2008 Quality management systems - Requirements. International Organization for Standardization. (2008)
31. Project Management Institute: A Guide to the Project Management Body of Knowledge (PMBOK Guide). (2008)
32. ISO/IEC: ISO 27001 Information technology - Security techniques - Information security management systems - Requirements. (2005)
33. García-Horcajadas, M.T., Ruiz-Carreira, M.: Alternativas para la transformacion de modelos de negocio a modelos de simulacion: Ontologias o MDE. In: Talleres de trabajo, Spain, JISBD (2009)
34. Escalona, M.J., Jiménez, A., Sánchez, N., Librada, S.: Servicio continuado de seguimiento de salud basado en tecnologias semanticas e inteligencia artificial enmarcadas en un entorno web 3.0. In: I Reunion de la Plataforma Tecnologica para la innovacion en la salud, Spain. (2012)