Un Entorno ALM Flexible y Dirigido por Procesos: El Proyecto Alcuza*

Amador Durán[1], Bernardo Panduro[2], Beatriz Bernárdez[1], and Juan D. Pérez–Jiménez[1]

[1] Dpto. de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla
{amador,beat,jdperez}@us.es
[2] Dpto. de Tecnología de Información de Cajasol
bpanduro@cajasol.es

Resumen Recientemente, el concepto de ALM (*Application Lifecycle Management*) ha surgido como una evolución de los desarrollos previos en *integración de herramientas* (*tool integration*) e *ingeniería del software centrada en procesos* (*process–centered software engineering*). La experiencia obtenida en el Departamento de Tecnología de Información de Cajasol ha puesto de manifiesto que el hecho de contar con entornos ALM dirigidos por procesos favorece la implantación de metodologías basadas en recomendaciones como CMMI–DEV, y por otro lado también facilita a los desarrolladores la utilización de las correspondientes herramientas de forma coordinada con los procesos prescritos metodológicamente. En su política de mejora de calidad de procesos internos, Cajasol decidió acometer el desarrollo conjunto de una serie de procesos metodológicos acordes al nivel 2 de madurez de CMMI–DEV, junto con un entorno ALM que facilitara su implantación en la entidad, dando lugar al proyecto *Alcuza*. La decisión de acometer el desarrollo de un sistema tan complejo como un ALM vino motivada por la falta de integración real de las herramientas comerciales disponibles, y por la idea de facilitar la adopción de los procesos metodológicos por parte del personal involucrado al estar soportados directamente por el entorno ALM. En este artículo se describen los problemas que promovieron la decisión de acometer el proyecto *Alcuza* y las principales características del entorno ALM desarrollado, haciendo especial énfasis en la integración en el mismo de los procesos de desarrollo mediante flujos de trabajo (*workflows*).

Keywords: Application Lifecycle Management, ALM, Integración de herramientas, Proceso de desarrollo, Workflow, CMMI.

1. Introducción

La cada vez mayor complejidad del software y los cada vez menores márgenes de tiempo para su entrega, hacen que la presión por incrementar la productividad y la calidad en las empresas de desarrollo de software sea cada vez mayor. Esta presión se ve aumentada también por la demanda cada día más habitual de certificaciones como

* Este trabajo ha sido financiado principalmente por el proyecto Alcuza de Cajasol a través de FIDETIA (P030–08/E15), y de forma parcial por los proyectos SETI (TIN2009–07366), THEOS (TIC–5906) e ISABEL (P07–TIC–2533).

CMMI–DEV [8] o similares que avalen a priori la capacidad de la empresa ante sus posibles clientes y la diferencien de su competencia. En este entorno altamente competitivo, una posible solución pasa por conseguir integrar herramientas y procesos en lo que se conoce actualmente como entornos ALM (*Application Lifecycle Management*) [18], cuyos conceptos básicos, junto con una revisión del trabajo relacionado de *integración de herramientas* y *desarrollo dirigido por procesos*, se ofrecen a continuación.

1.1. Conceptos básicos de ALM

El concepto de ALM se populariza entre los fabricantes de herramientas de desarrollo de software a partir del informe de la consultar Forrester en 2006 [18], en el que se propone una evolución de los entornos de desarrollo habituales hacía entornos basados en los denominados tres *pilares* del ALM [18]:

1. **Automatización de los procesos de alto nivel.** Los procesos a seguir durante el desarrollo de software no deben ser libros que permanecen en las estanterías sin que nadie los use ni tampoco trabas burocráticas para los desarrolladores. Los procesos deben poder ser ejecutables.
2. **Visibilidad del progreso de los esfuerzos de desarrollo.** Los gestores de proyecto deben ser capaces de ver la evolución de los proyectos para tomar decisiones, normalmente mediante *cuadros de mando*, o mediante informes más detallados cuando sea necesario, que muestren los indicadores claves de proceso.
3. **Trazabilidad de las relaciones entre artefactos.** Aunque reconocido como algo esencial desde hace muchos años, la trazabilidad total entre los distintos artefactos que se producen en el desarrollo de software está aún lejos de ser posible, principalmente debido a la falta de integración de las herramientas.

Para alcanzar estos objetivos, en el propio informe de Forrester y en [2] se proponen las siguientes características para una arquitectura de un entorno ALM:

- Herramientas ensambladas a base de *plugins*, que permitan un entorno flexible y ampliable, con una interfaz de usuario común, que facilite el aprendizaje y favorezca el trabajo en equipo.
- Independencia de repositorio, metamodelos compartidos, servicios comunes y uso de estándares de integración abiertos, que faciliten el desarrollo de herramientas, su interoperabilidad y la trazabilidad entre los artefactos.
- *Micro* y *macroprocesos* gobernados por un motor de *workflow*, que ayude a los desarrolladores a seguir los procedimientos establecidos.

1.2. Desarrollo de software dirigido por procesos

La idea de modelar y automatizar el proceso de desarrollo de software no surge con el concepto de ALM, sino de trabajos a partir de la segunda mitad de la década de 1980, siendo el más conocido el clásico artículo de Osterweil [14]. Pese a los esfuerzos realizados desde entonces, ni los resultados obtenidos ni su transferencia a la industria han sido los esperados, debido a que el proceso de desarrollo de software es un proceso complejo que presenta ciertas características como cambios continuos, evolución

constante, participación de un gran número de personas, etc. que hacen que, en muchas ocasiones, no pueda ser completamente definido a priori, tal como se discute en [4].

Dentro de los modelos de proceso de software se distinguen dos posibles tipos según [7,13], que pueden ser:

- **Modelos proscriptivos**: son modelos de procesos *laxos* en los que se permite cualquier tipo de actividades siempre que se cumplan una serie de restricciones, normalmente expresadas mediante reglas.
- **Modelos prescriptivos**: son modelos de procesos más rígidos en los que se especifican detalladamente las acciones a desarrollar indicando además el orden en el que deben ejecutarse, normalmente expresado mediante la utilización notaciones gráficas como SPEM [17] o similares.

En general, el concepto de ALM se orienta más hacia modelos prescriptivos, pero con cierta flexibilidad que permita la ejecución de procesos *semi–modelados* o *ad hoc* sobre motores de *workflow*, como se comenta en [10]. Sin embargo, cierta flexibilidad en la definición de los procesos no es suficiente, ya que también es relevante el problema de la integración de los *micro* y *macroprocesos* [15,21,22], es decir, entre los procesos de bajo nivel que suelen seguir un modelo prescriptivo (p.e. gestionar una petición de cambio) y los de alto nivel asociados al ciclo de vida elegido para el desarrollo y que están más cerca del modelo proscriptivo. Las metodologías actuales no suelen incluir los microprocesos, pero según [22], deberían tratarse con más detalle para mejorar la comprensión de las relaciones causales en el desarrollo de software y para mejorar la visión general del proceso por parte de los desarrolladores. De hecho, en [5] se reconoce como uno de los peligros de este tipo de enfoques la utilización de modelos de procesos demasiado genéricos que no ayuden al desempeño de las tareas diarias.

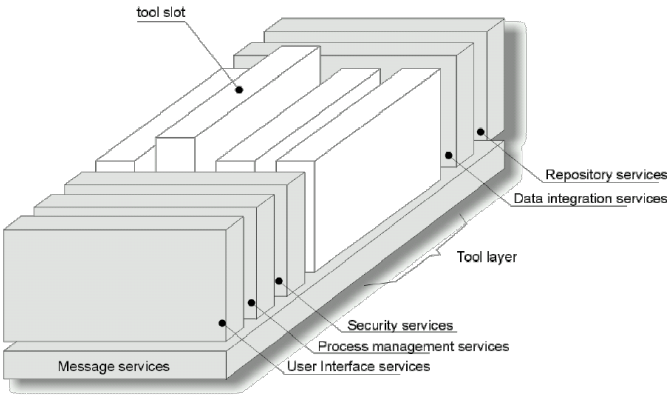
1.3. Integración de herramientas de ingeniería del software

Al igual que el desarrollo dirigido por procesos, la necesidad de integración de herramientas es anterior al concepto de ALM, como se establece el artículo clásico de Wasserman [19]: "*a key issue in CASE environments is the desire to link tools that address different aspects of the development process*". Sin embargo, a pesar del tiempo transcurrido, la integración de herramientas sigue siendo un problema para el que aún no se han encontrado soluciones en la industria con éxito comercial [12,20].

Los diferentes enfoques que se han ido planteando desde la década de 1980 han propuesto soluciones puramente tecnológicas y poco flexibles, sin tener en cuenta aspectos socio–económicos [20]. Esto ha provocado que ni los enfoques iniciales basados en entornos centrados en el proceso (PCSEEs, *Process–Centred Software Engineering Environments*) como SPADE–1 [6] o PRO–ART [16], ni otros más recientes basados exclusivamente en la integración de datos sobre XML/XMI hayan terminado de asentarse en la industria. Un hecho claramente significativo de esta situación es la inexistencia de herramientas que implementen un estándar desarrollado expresamente para la integración de herramientas CASE como el IEEE 1175.1 [1].

Un resultado de interés de estos trabajos es el modelo de referencia para *frameworks* de entornos de ingeniería de software conocido como "*the toaster*" (la *tostadora*, ver

Figura 1). Propuesto en 1989 por Earl [11], y posteriormente adoptado por la ECMA (*European Computer Manufacturer's Association*), dicho modelo integra muchas de las características deseables de la arquitectura para entornos ALM comentadas en la Sección 1.1. Es interesante ver como, a pesar de la fecha en que se propuso, muchas de sus características son relativamente fáciles de implementar hoy en día sobre una arquitectura orientada a servicios (SOA, *service–oriented architectures*, que es precisamente la elegida para el proyecto *Alcuza*.



Fuente del diagrama: `http://www.softwarekompetenz.de/?22364&highlight=konkretes`

Figura 1. Modelo de referencia ECMA ("*the toaster*")

1.4. Estructura del artículo

El resto del artículo se organiza como se describe a continuación. La Sección 2 describe los antecedentes del departamento de Tecnologías de Información (TI) de Cajasol que determinaron la decisión de acometer el proyecto *Alcuza*. Las Secciones 3 y 4 describen respectivamente la arquitectura de integración de herramientas y la orientación a procesos en el ALM Alcuza. Finalmente, en la sección 5, se presentan las conclusiones y los posibles trabajos futuros.

2. Antecedentes del proyecto Alcuza

El software esencial de Cajasol[3] comparte una serie de características comunes con otras entidades financieras, principalmente: (1) está desarrollado en COBOL sobre el servidor de transacciones CICS [3]; (2) su antigüedad media ronda los 20 años; y (3) la mayor parte del esfuerzo se centra en mantenimiento más que en desarrollo de nuevo software. Por otro lado, el departamento de TI, encargado del desarrollo y mantenimiento de software, está formado principalmente por jefes de proyecto y analistas, mientras

[3] Es decir, el software que da soporte a las operaciones financieras, su principal negocio.

que las labores de programación están externalizadas. Hasta la fusión que dio lugar a la actual Cajasol en 2006, en la externalización se contrataban horas/hombre que se consumían bajo demanda. Tras la fusión, se pasó a contratar trabajos completos y no horas/hombre, lo que generó la necesidad de gestionar dichos trabajos como proyectos y el desarrollo de varias herramientas internas. La principal fue *Astarté*, un gestor de peticiones/tareas (ver Figura 2), que permitía a cualquier empleado de Cajasol solicitar o reportar incidencias, peticiones de trabajo, peticiones de cambio, propuestas de proyectos o peticiones internas al departamento de TI, que las gestionaba como tareas, cada una con su propio *microproceso* definido y gestionado por la herramienta.

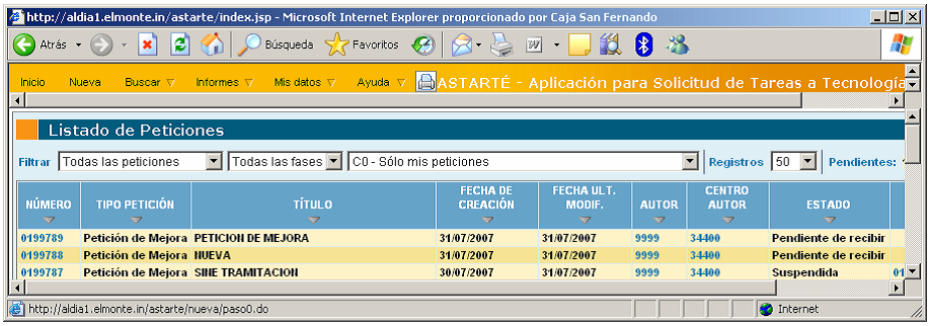


Figura 2. Interfaz web de la herramienta Astarté de Cajasol

A pesar de que la implantación de Astarté supuso una mejora importante, el volumen de trabajo cada vez mayor asociado al crecimiento de la entidad por las fusiones puso de manifiesto una serie de problemas importantes, algunos de ellos debido a la falta de cultura de gestión de proyectos de TI:

- No se conocía el grado de ocupación de los recursos humanos del departamento de TI ni su dedicación a los distintos proyectos, resultando imposible proporcionar fechas estimadas de finalización de trabajos a los peticionarios.
- Los canales de comunicación con los departamentos centrales (clientes internos) no estaban formalizados, por lo que no siempre quedaba registro de las peticiones realizadas al departamento de TI y era muy difícil seguir su estado.
- No había uniformidad a la hora de realizar los trabajos de desarrollo. Cada jefe de proyecto aplicaba las técnicas de gestión que creía oportuno y generaba la documentación que estimaba pertinente con el formato que necesitase en cada momento.

En resumen, existían cientos de peticiones y miles de tareas asociadas, pero no se sabía en qué situación estaban los trabajos ni las peticiones prioritarias para el negocio. No se usaba el concepto de proyecto, es decir, no se agrupaban las peticiones con un fin común, por lo que la misma petición podía originar distintas tareas independientes una de otra con distintos responsables. En esta situación, el seguimiento y control por los peticionarios era prácticamente imposible y la gestión de los recursos de TI una labor muy difícil.

2.1. El origen del proyecto Alcuza

Para resolver los problemas descritos, se toma la decisión estratégica de implantar en el departamento de TI el modelo CMMI–DEV [8] para la gestión de proyectos de desarrollo y mantenimiento de software hasta el nivel 2, incluyendo además las *áreas de proceso* de nivel 3 de *Desarrollo de Requisitos* (RD) y *Gestión de Riesgos* (RSKM).

Gracias a la experiencia obtenida durante la implantación del gestor de peticiones *Astarté*, los responsables del departamento de TI eran conscientes de: (1) la gran resistencia al cambio dentro del departamento de TI de Cajasol; (2) los beneficios de las sinergias *procedimiento–workflow–herramienta* (ver Figura 3), es decir implantar procedimientos mediante herramientas software que *obligan* al personal involucrado seguir mediante un *workflow* los procedimientos prescritos, los cuales a su vez también *imponen* usar las correspondientes herramientas software; (3) la familiaridad del personal del departamento de TI con herramientas soportadas por *workflows* como la de tramitación de riesgos o la anteriormente citada *Astarté*.

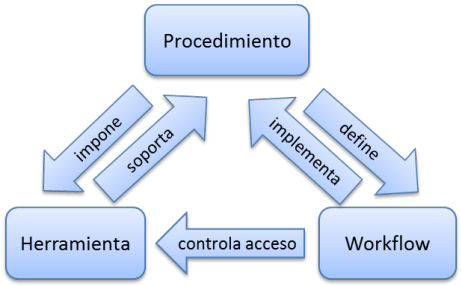


Figura 3. Sinergias procedimiento–workflow–herramienta

Por otro lado, el departamento de TI de Cajasol elaboró un informe de evaluación de herramientas de ingeniería del software en el que se analizaron tanto su capacidad de integración como las facilidades que presentaban para dar soporte a los procesos de CMMI–DEV hasta nivel 2, con especial hincapié en las áreas de procesos que suelen presentar más problemas para su implantación:[4] REQM (*Requirements Management*), RD (*Requirements Development*), PP (*Project Planning*), PMC (*Project Monitoring and Control*), CM (*Configuration Management*), VER (*Verification*) y VAL (*Validation*). Los resultados de este informe (ver Tabla 1), corroborando la experiencia del departamento de TI de Cajasol, indicaron que prácticamente ninguna de las herramientas evaluadas proporcionaban una integración efectiva; casi ninguna daba soporte de manera explícita a procesos acordes a CMMI–DEV; y, probablemente lo más relevante, el impacto de su implantación en la entidad era muy alto, obligando a los desarrolladores a adaptarse a las herramientas en lugar de adaptar las herramientas a los desarrolladores, por lo que se preveía un fracaso en su su implantación.

[4] Se incluyen RD, VER y VAL porque, aunque son de nivel 3, suelen abordarse también al intentar alcanzar el nivel 2 y era intención de Cajasol abordarlas desde un principio.

Criterios de evaluación	Peso	Microsoft EPM	IBM Rational	Borland ALM	Serena	Team Vision	Polarion
Coste de la herramienta	7	1	1	1	1	2	3
Dificultad de implantación	7	3	2	2	1	3	2
Coste de recursos de implantación	5	1	0	0	0	2	2
Compatibilidad con las aplicaciones existentes en Cajasol	8	3	2	1	1	3	1
Cobertura de necesidades	10	1	3	2	2	1	2
Total evaluación (máx. 148)	—	**67**	**67**	**49**	**42**	**79**	**73**

Tabla 1. Resultados del informe de evaluación de herramientas de Cajasol

Teniendo en cuenta estas consideraciones, la dirección del departamento de TI consideró esencial que, en lugar de implantar CMMI–DEV como un conjunto de procedimientos soportados por un grupo de herramientas separadas o aisladas (el clásico problema de la *integración de herramientas*), se desarrollara un entorno ALM en el que se integraran todas las herramientas necesarias, algunas ya existentes y otras de nuevo desarrollo, y que estuviera dirigido por un motor de workflow que *obligara* a sus usuarios a seguir los procedimientos acordes a CMMI–DEV que se decidiera implantar.

3. La arquitectura de Alcuza

Inspirados por el modelo ECMA (ver Figura 1), Alcuza se planteó desde un primer momento con una arquitectura SOA sobre una plataforma Java interna de Cajasol denominada Ateneo (ver Figuras 4 y 5) que, entre otras características, permite exponer funcionalidades preexistentes como servicios, incluyendo incluso las transacciones financieras desarrolladas en COBOL. Esto permite reutilizar gran parte de las aplicaciones desarrolladas previamente y que las nuevas puedan exponer su funcionalidad como servicios de forma relativamente sencilla, reduciendo así los costes de integración. Ateneo también proporciona una serie de servicios horizontales como seguridad, auditoria, monitorización, estadísticas, etc. que simplifican el desarrollo de nuevas aplicaciones.

3.1. Principales componentes de Alcuza

En este contexto, podemos definir Alcuza como una aplicación con una interfaz de usuario RIA (*Rich Internet Application*) desarrollada en Adobe Flex y una arquitectura SOA en servidor desplegada sobre la plataforma Ateneo. Su principales componentes (ver Figura 5) son el *gestor de actividades* y el *gestor de configuración*, además de otros construidos sobre ellos como los gestores de *requisitos*, *riesgos* o *problemas*.

El gestor de configuración controla el acceso a los repositorios, actualmente dos, uno para código y otro para documentación, pero que se unificarán en un futuro próximo. El control implica que ningún usuario puede acceder a documentos o a código

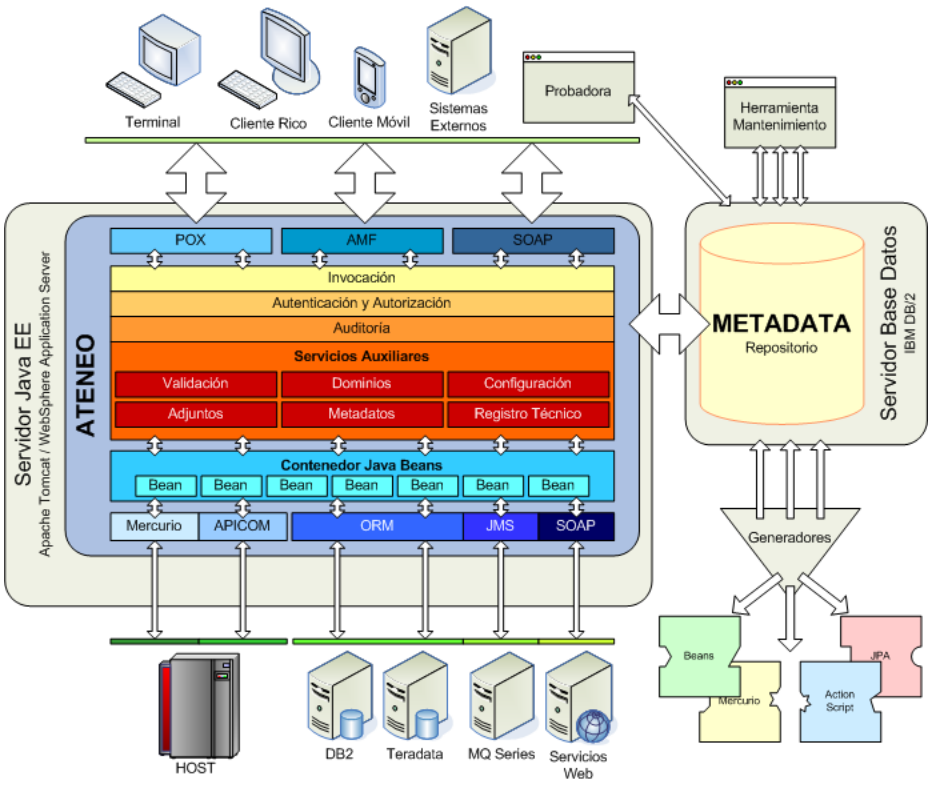


Figura 4. Arquitectura de la plataforma Ateneo de Cajasol

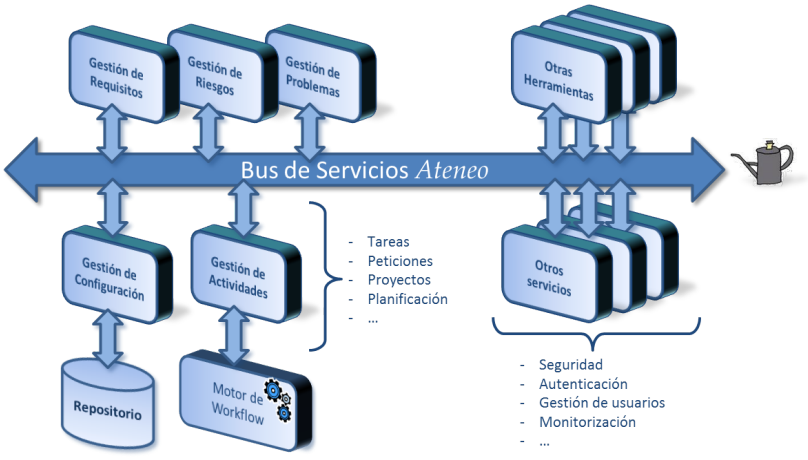


Figura 5. Arquitectura orientada a servicios de Alcuza

fuente sin que haya una actividad en el gestor de actividades que le permita hacerlo, reforzando de esta forma la sinergia *procedimiento–workflow–herramienta* comentada en la sección 2.1. Además, implementa una política de versionado automático que al terminar una actividad crea automáticamente una nueva versión de los artefactos que se hayan modificado, incluyendo la posibilidad de crear *líneas base* a nivel de proyecto.

Por otro lado, el gestor de actividades implementa la orientación a procesos de Alcuza y es su núcleo central. Aparte de mantener la información correspondiente a todas las actividades de cualquier proyecto de Cajasol, dispone de un motor de workflow interno cuyo funcionamiento se describe en la siguiente sección.

Sobre los servicios proporcionados por estos dos componentes se han ido desarrollando e integrando herramientas que soportan los procedimientos metodológicos acordes a CMMI–DEV. Así, la primera fue el *gestor de requisitos*, desarrollado a partir de las ideas implementadas en la herramienta REM [9] desarrollada en la Universidad de Sevilla, e incorporando facilidades para los reutilización como plantillas de documentos y patrones de requisitos. Posteriormente, una vez definidos los procedimientos acordes a CMMI–DEV, se fueron desarrollando sus herramientas asociadas como el cartera de proyectos, la agenda de tareas, el gestor de riesgos, el de problemas, etc.

4. La orientación a procesos en Alcuza

Alcuza sigue un modelo de procesos *prescriptivo* pero flexible, que permite añadir actividades *ad hoc* al usuario responsable de una actividad según sus necesidades. De esta forma, a partir de un modelo de procesos de alto nivel (*macroproceso*), los usuarios pueden refinarlo hasta que lo consideren necesario (*microproceso*), pero siempre con una estructura jerárquica, evitando así el hueco entre ambos niveles de detalle [15].

Figura 6. Propiedades básicas de una actividad en Alcuza

4.1. El concepto de actividad en Alcuza

En Alcuza, la orientación a procesos se basa en el concepto de *actividad*, que se considera como *el conjunto de acciones unitarias que realiza un individuo para conseguir un fin*. Esta definición implica que una actividad puede descomponerse en otras, y que va a estar asignada a un único responsable que supervise su evolución o la realice.

Las actividades en Alcuza tienen propiedades básicas como un título, una descripción, datos de planificación, un empleado responsable, etc. (ver Figura 6). Además, en Alcuza se han añadido tres *aspectos* que, al poder combinarse, permiten la flexibilidad necesaria y que son el *modelo (de ciclo de vida)*, el *tipo de actividad* y el *tipo de trabajo*.

4.2. Los aspectos de las actividades en Alcuza

Modelo de ciclo de vida El primer aspecto de una actividad Alcuza es el *modelo de ciclo de vida*, cuyo esquema básico es un ciclo *en tramitación–en curso–terminada* como el que puede verse en el diagrama de estados de la Figura 7, aunque los distintos modelos pueden añadir subestados si es necesario. Actualmente, Alcuza incluye tres modelos predefinidos: *completo*, que es el seguido por todas las peticiones de trabajo; *directo*, orientado a la descomposición de actividades y sin estado de tramitación; y *de etiquetado*, usado para las actividades intermedias que se usan para indicar hitos y que sólo pueden estar *etiquetadas* como abiertas (*en curso*) o cerradas (*terminadas*).

Las transiciones de un estado a otro son controladas por el gestor de actividades, de forma que sólo se permite dispararlas a los usuarios con lo suficientes privilegios cuando la actividad se encuentra en el estado correcto y con los datos correspondientes cumplimentados. Por ejemplo, ninguna actividad puede terminarse hasta que todas sus actividades hijas lo hayan hecho. Todas aquellas acciones del usuario que no se puedan realizar por alguna causa, aparecerán deshabilitadas, mostrando así una visión más clara de qué se puede y qué no se puede hacer en cada momento.

Tipo de actividad El tipo de actividad es el aspecto más importante de una actividad, ya que, a parte de permitir añadir nuevas propiedades, va a determinar su *modelo de procesos*, es decir, su descomposición en actividades hijas y la secuenciación de las mismas, que será controlada por el motor de workflow integrado en el gestor de actividades. De esta forma, conforme se vayan cerrando unas actividades se irán inyectando las siguientes según el modelo de procesos asociado. Actualmente, Alcuza incluye varios tipos de actividad predefinidos (*mejora*, *cambio*, *incidencia*, *fase*, *actividad interna*,

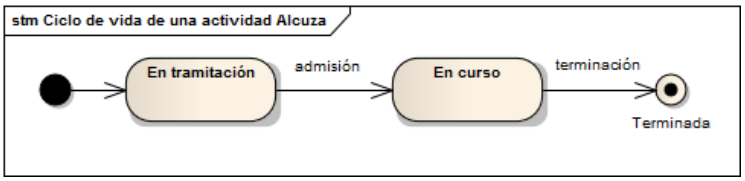


Figura 7. Ciclo de vida básico de una actividad en Alcuza

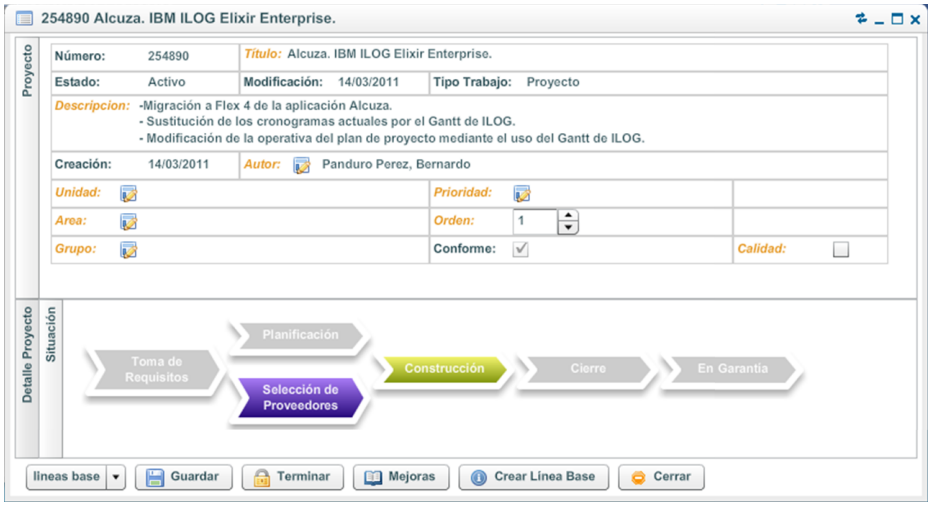


Figura 8. Detalle de un proyecto Alcuza con sus fases

etc.), siendo el más relevante de ellos el *proyecto* (ver Figura 8), con modelo de ciclo de vida *completo*, con un modelo de procesos detallado (ver Figura 9) y que incluye como propiedad su *organigrama* (ver Figura 10), que es la forma en que Alcuza incluye el modelado organizacional asociado al modelado de procesos.

Tipo de trabajo Este tercer aspecto permite especificar las entradas y salidas de una actividad, y por tanto a qué herramientas puede acceder el usuario responsable. También permite indicar si la actividad necesita seguimiento u otros aspectos de calidad como pasar una lista de comprobación al terminarla (ver Figura 11). Actualmente, Alcuza incluye cuatro tipos de trabajo, relacionados con documentación de requisitos, codificación y pruebas, seguimiento y control de proyectos y trabajos genéricos.

5. Conclusiones y trabajo futuro

Aunque por ahora no se dispone de resultados cuantitativos que permitan demostrar si se han logrado alcanzar los objetivos planteados al comienzo, sí se observa que Alcuza ha traído consigo una mejoría notable en la forma de trabajar del departamento de TI de Cajasol, principalmente por las sinergias *procedimiento–workflow–herramienta* comentadas en la Sección 2.1. Alcuza se ha convertido en una guía para saber qué trabajo hay que hacer y cuándo, qué documentos son obligatorios, qué información hay que aportar en cada momento, etc., facilitando enormemente la adopción de los procedimientos metodológicos elaborados para seguir las recomendaciones de CMMI–DEV. Por otro lado, la imposibilidad de llevar a cabo las actividades sin usar Alcuza permite la visibilidad del estado de los proyectos al estar toda la información de la planificación y el esfuerzo centralizada, por lo que los responsables pueden tener en cada momento una visión global y a la vez detallada del estado de los proyectos.

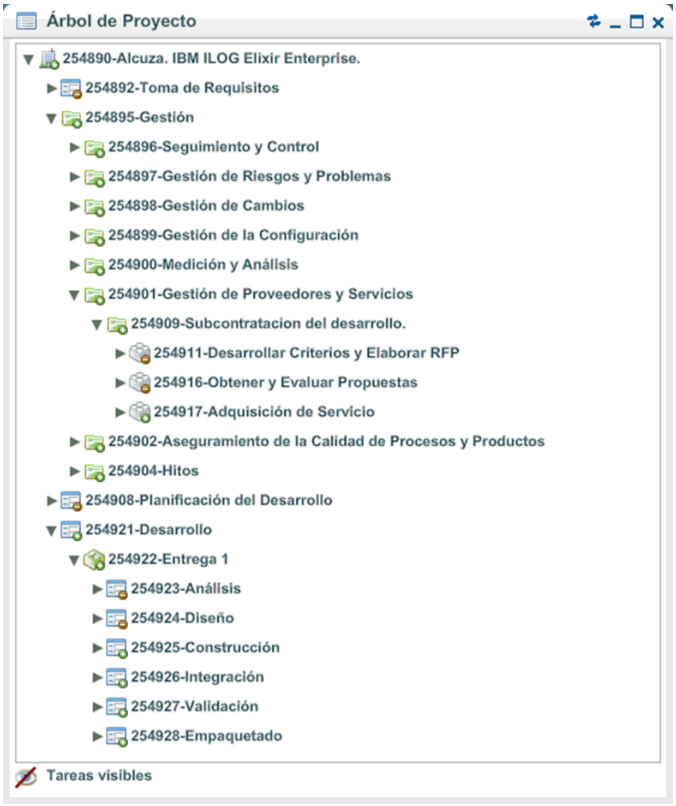


Figura 9. Árbol de actividades de un proyecto Alcuza

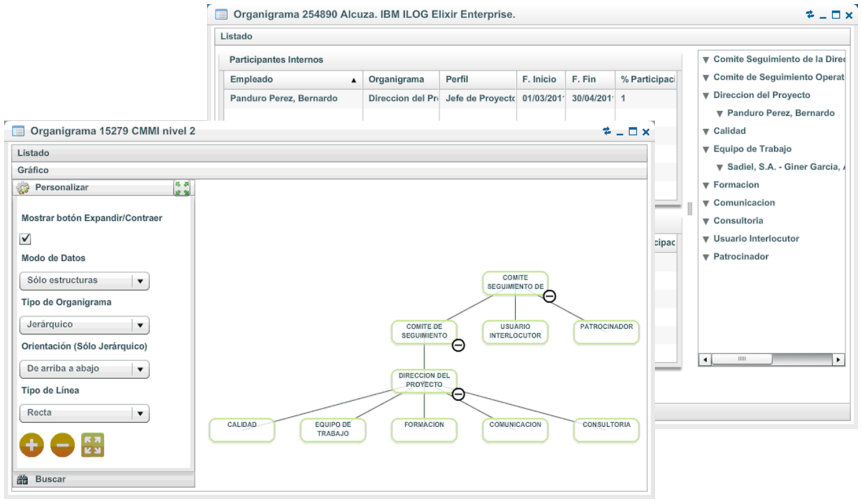


Figura 10. Organigrama de un proyecto Alcuza (vistas gráfica y en lista)

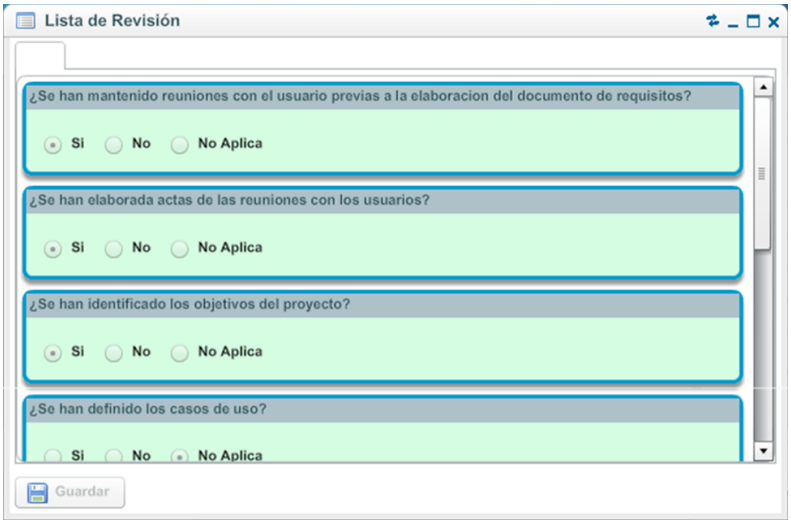


Figura 11. Lista de comprobación asociada a tipo de trabajo de documentación de requisitos

En cuanto a la certificación CMMI–DEV, para mediados de junio de 2011 se ha planificado la evaluación final (SCAMPI A) para conseguir la acreditación del nivel 2. En todas las evaluaciones intermedias que se han realizando a lo largo del año pasado y del actual, la mayor fortaleza constatada por los consultores que han visitado la entidad ha sido la herramienta Alcuza.

Como trabajo futuro, se ha planteado la posibilidad de realizar un estudio de campo para comparar cualitativamente proyectos *pre–Alcuza* con proyectos *post–Alcuza*. Para ello, podría resultar interesante definir un conjunto de indicadores de calidad de los procesos, recursos o productos de desarrollo y analizar en qué sentido Alcuza mejora dichos elementos. Otra técnica más directa, aunque con resultados menos objetivos, podría ser realizar encuestas mediante entrevistas o cuestionarios para conocer la percepción que los usuarios tienen de la nueva forma de trabajar y del nuevo entorno de trabajo (teniendo en cuenta la resistencia al cambio, innata en el ser humano). Esto también debe permitirnos conocer las carencias del ALM y las mejoras que proponen los usuarios para el mismo.

Agradecimientos

A Antonio Barros Carmona, exdirector del departamento de TI de Cajasol (actualmente prejubilado), por ser el principal impulsor del proyecto Alcuza y el origen de la mayoría de las ideas que incorpora.

Referencias

1. IEEE Guide for CASE Tool Interconnections - Classification and Descriptions. IEEE Std 1175.1–2002 (2003), revisión de IEEE Std 1175.1–1991

2. Open Application Lifecycle Management (ALM). White paper, Borland Software Corporation (2007)
3. IBM CICS Family official website. http://www-01.ibm.com/software/htp/cics/ (2010)
4. Arbaoui, S., Derniame, J., Oquendo, F., Verjus, H.: A Comparative Review of Process–Centered Software Engineering Environments. Annals of Software Engineering 14(1–4), 311–340 (2002)
5. Armbrust, O., et al.: Scoping Software Process Models — Initial Concepts and Experience from Defining Space Standards. In: International Conference on the Software Process. LNCS, vol. 5007, pp. 160–172 (2008)
6. Bandinelli, S., et al.: Combining Control and Data Integration in the SPADE–1 Process–Centered Software Engineering Environment. In: 9^{th} International Software Process Workshop (1994)
7. Barthelmess, P.: Collaboration and Coordination in Process–Centered Software Development Environments: A Review of the Literature. Information and Software Technology 45(13), 911–928 (2003)
8. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI: Guidelines for Process Integration and Product Improvement. Addison–Wesley, 2^{nd} ed. edn. (2006)
9. Durán, A.: Sitio web de la herramienta REM. http://www.lsi.us.es/descargas/descarga_programas.php?id=3 (2011)
10. Dustdar, S.: Caramba—A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams. Distributed and Parallel Databases 15(1), 45 – 66 (2004)
11. Earl, A.: Principles of a Reference Model for Computer Aided Software Engineering Environments. In: Software Engineering Environments. LNCS, vol. 467, pp. 115–129. Springer (1990)
12. Gruhn, V.: Process–Centered Software Engineering Environments, A Brief History and Future Challenges. Annals of Software Engineering 14(1-4), 363 – 382 (2002)
13. Heimbigner, D.: Proscription versus Prescription in Process–Centered Environments. In: 6^{th} International Software Process Workshop. pp. 99–102 (1990)
14. Osterweil, L.: Software Processes are Software Too. In: 9th International Conference on Software Engineering (1987)
15. Osterweil, L.: Unifying Microprocess and Macroprocess Research. In: Unifying the Software Process Spectrum, LNCS, vol. 3840, pp. 68–74 (2006)
16. Pohl, K., Weidenhaupt, K.: A Contextual Approach for Process–Integrated Tools. ACM SIGSOFT Softw. Eng. Notes 22, 176–192 (1997)
17. Ruiz, F., Verdugo, J.: Guía de Uso de SPEM 2 con EPF Composer (versión 3.0). http://alarcos.inf-cr.uclm.es/doc/psgc/doc/lec/parte2b/guia-spem2&epf_v30.pdf (2008)
18. Schwaber, C.: The Changing Face Of Application Life–Cycle Management. Tech. rep., Forrester Research, Inc. (2006)
19. Wasserman, A.I.: Tool Integration in Software Engineering Environments. In: Software Engineering Environments. LNCS, vol. 467, pp. 137–149 (1990)
20. Wicks, M.N., Dewar, R.G.: A New Research Agenda for Tool Integration. Journal of Systems and Software 80(9), 1569–1585 (2007)
21. Zhu, L., Jeffery, R., Staples, M., Huo, M., Tran, T.: Effects of Architecture and Technical Development Process on Micro–process. In: Software Process Dynamics and Agility, LNCS, vol. 4470, pp. 49–60 (2007)
22. Zhu, L., Tran, T., Staples, M., Jeffery, R.: Technical Software Development Process in the XML Domain. In: Trustworthy Software Development Processes, LNCS, vol. 5543, pp. 246–255 (2009)