Revista Española de Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 6, No. 2, octubre, 2010

Web de la editorial: www.ati.es
Web de la revista: www.ati.es/reicis
E-mail: calidadsoft@ati.es
ISSN: 1885-4486

Publicado por la Asociación de Técnicos de Informática (ATI), Via Laietana, 46, 08003 Barcelona.

Secretaría de dirección: ATI Madrid, C/Padilla 66, 3º dcha., 28006 Madrid

Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS)

Contenidos

REICIS

Definición de una política de pruebas en la gestión cultural: aplicación al desarrollo del proyecto Mosaico

J. Ponce, M.J. Escalona,
Universidad de Sevilla
{ josepg , mjescalona}@us.es
A. Gómez, M. Luque
Empresa Pública para la Gestión de Programas Culturales.
Consejería de Cultura. Junta de Andalucía
{antonio.gomez.ro, manuel.luque.ramos}@juntadeandalucia.es
A. Molina
Consejería de Cultura. Junta de Andalucía
antonio.molina.gonzalez@juntadeandalucia.es

Resumen

Definir un contexto para la gestión de políticas de pruebas en un contexto práctico no es siempre sencillo. A pesar que de que en la bibliografía podemos encontrar diferentes referentes de pruebas a la hora de implementar estos modelos teóricos se encuentran problemas y deben resolverse un gran número de aspectos operativos. Este artículo, realizado en colaboración entre la universidad de Sevilla y la Consejería de Cultura, describe cómo una propuesta teórica ha sido llevada a la práctica en Mosaico, un sistema de desarrollo complejo. El artículo presenta la implementación realizada bajo un perfil y una serie de políticas de trabajo llevadas a cabo con los usuarios.

Palabras Clave: Pruebas, Administración Cultural

Defining a policy of tests in cultural management: application to the development of the Mosaico project

Abstract

To define a practical context for the management of policies of tests is not simple. Although we can find different models of tests in the bibliography, there are problems when theoretical models have to be implemented and many operative aspects must be solved. This article, which is realized between the University of Seville and the Council of Culture, presents how a theoretical offer has been taken to the practice in a complex system, named Mosaico. The article presents how the offer has been implemented under a profile and several policies of work carried out with the users.

Key words: Testing, Cultural Government

*Ponce, J.. Escalona,M.J., Gómez, A., Luque, M. y Molina, A.. "Definición de una política de pruebas en la gestión cultural: aplicación al desarrollo del proyecto Mosaico", REICIS, vol. 6, no.2, 2010, pp.26-**. Recibido: 19-5-201; revisado: 25-8-2010; aceptado: 19-9-2010.*

1. Introducción

La fase de pruebas es una de más relevantes a la hora de desarrollar software. El problema es que, en muchos casos, los requisitos de costes y de plazos implican que no siempre se puedan abordar los trabajos planificados en las pruebas. El control de calidad de cualquier producto debe comprobar que la calidad del producto final se ajusta a una especificación dada. En el caso concreto del software, el control de calidad debe comprobar que el producto final funcione correctamente, de acuerdo con sus especificaciones, y en colaboración con otros sistemas software y bases de datos [1].

Un aspecto crucial de control de la calidad son las pruebas que se aplican al producto final, las cuales permiten detectar en qué puntos dicho producto no satisface sus especificaciones. El objetivo de estas pruebas es probar a fondo el sistema, comprobando su funcionalidad e integridad globalmente, en un entorno lo más parecido posible al entorno final de producción.

Este tipo de pruebas, aplicadas a los sistemas software, se conocen como pruebas de sistema y su objetivo es comprobar que el sistema software que se está desarrollando cumple con la funcionalidad recogida en casos de uso o escenarios. La planificación de estas pruebas y el diseño de los casos de prueba a los que someteremos al sistema deben comenzar tan pronto como estén disponibles las especificaciones funcionales. La planificación y diseño de pruebas en las primeras fases de desarrollo permiten encontrar errores, omisiones, inconsistencias e, incluso, especificaciones redundantes en los requisitos funcionales cuando aún es fácil y económico corregirlas ya que el coste de eliminar defectos aumenta a medida que aumenta el tiempo qué transcurre entre la aparición del defecto y su detección [1]. Para que el proceso de prueba del sistema sea eficaz debe estar integrado dentro del propio proceso de desarrollo y, como cualquier otra fase del desarrollo, debe realizarse de manera sistemática.

En este artículo, se presenta cómo se han abordado las pruebas en un sistema denominado Mosaico [2]. Este sistema que permite gestionar y difundir la información sobre el patrimonio cultural se caracteriza por ser un sistema complejo y grande que se desarrolla en unas bases tecnológicas difíciles.

El artículo se estructura de la siguiente manera. Comienza con una revisión de los antecedentes y trabajos relacionados con la temática. Tras esto, se presenta una visión

general de Mosaico y se continúa con una presentación de cómo se ha abordado la fase de pruebas de Mosaico. El artículo termina con las conclusiones y trabajos futuros.

2. Antecedentes y contexto

Dentro de las áreas de actuación de la Consejería de Cultura, la Dirección General de Bienes Culturales (DGBC) tiene la responsabilidad de gestionar y documentar, difundir y proteger el Patrimonio Cultural de Andalucía. Este objeto de trabajo se ha venido realizando por los diferentes centros y servicios de la DGBC de forma normalizada o unificada. Tras el desarrollo del plan de sistemas de información 2000-2004 de la Consejería de Cultura, se analizó la necesidad de dotar de las ventajas que ofrecen las tecnologías de la información a esta gestión del patrimonio.

Mosaico es un sistema global y horizontal en la DGBC que va orientado a conseguir los siguientes objetivos:

- Ofrecer los recursos y las herramientas tecnológicas necesarias para gestionar la información del Patrimonio Histórico.
- Ofrecer un sistema de información global que almacenará la información y documentación, incluida la georreferencial, de todo el Patrimonio Cultural de Andalucía.
- Facilitar el acercamiento del público en general a la Administración, y más concretamente a la información sobre el Patrimonio. Basado en las tecnologías y plataformas suministradas por la Administración Electrónica, Mosaico debe ofrecer la posibilidad de consultar información sobre el Patrimonio Andaluz.

Abordar un proyecto de la complejidad de Mosaico requiere que se apliquen técnicas adecuadas de ingeniería del software que garanticen la calidad de los resultados. Como se presenta en el siguiente apartado, Mosaico se ha abordado desde un punto metodológico dividiéndolo en diferentes fases para su trabajo. Este artículo está centrado a presentar como, llegados a la fase de pruebas, se ha abordado su trabajo.

Cuando se planteó realizar las pruebas funcionales, se ha hecho uso de la definición de los casos de uso del sistema. Este tipo de trabajo se está conociendo como early testing y consiste en intentar abordar el alcance de las pruebas en función de los requerimientos. En

el caso de Mosaico se ha usado la propuesta definida basada en la metodología NDT (Navigational Development Techniques)[3].

3. La complejidad del entorno de Mosaico

Mosaico es el Sistema de Información de Gestión de Bienes Culturales de la Consejería de Cultura de la Junta de Andalucía. Antes del planteamiento de Mosaico, los usuarios de la Consejería de Cultura encargados de la gestión del patrimonio cultural andaluz, debían utilizar varios sistemas para la gestión de la información sobre los bienes culturales y para registrar las actividades que llevan a cabo sobre estos bienes. Estos sistemas eran independientes y sin mecanismos de conexión, y dan lugar a unas tareas de consolidación y extracción de información muy costosas.

3.1. Alcance y objetivos de Mosaico

El objetivo general de Mosaico es la incorporación en un único sistema de toda la información y los procesos necesarios para permitir la gestión y conocimiento del Patrimonio Histórico de Andalucía, convirtiéndose en un instrumento para la mejora de la Protección y Difusión de estos elementos con valor patrimonial. Para ello integra y normaliza la información que ya se almacena en las aplicaciones existentes y completa sus funcionalidades para dar soporte a los procesos gestión de la Consejería de Cultura alrededor del Patrimonio Histórico. A continuación se analizan los puntos más importantes del sistema.

1. Modelo conceptual. La entidad principal alrededor de la que se orquestan los procesos de Mosaico son los bienes del Patrimonio Histórico, denominados como Objetos de Registro. Los distintos procesos que se ejecutan en Mosaico, además de completar datos de conocimiento del bien, aportan información en forma de documentos, expedientes administrativos e información geográfica. Para la ejecución de estos procesos es necesario tener una información normalizada en el ámbito de los actores y en el de la terminología utilizada.
2. Entidades externas. Fuera del sistema, Mosaico se nutre de la información y servicios de los sistemas de dos ámbitos diferentes: sistemas de administración electrónica de la Junta de Andalucía y sistema de información geográfica. De los primeros, Mosaico obtiene soporte para realizar toda la gestión de expedientes de forma electrónica; los

sistemas que utiliza son @firma, trewa, @ries, notific@, @visador, port@firmas. De los segundos, Mosaico es "consumidor" de mapas, disponibles mediante servicios web, en distintos servidores de mapas de dentro y fuera de la Junta (catastro, callejeros, etc).

3. Procesos. Mosaico facilita la gestión de toda la información que la actividad de la Consejería de Cultura genera sobre los bienes del Patrimonio Histórico.

En la siguiente figura se muestra un ejemplo de la arquitectura funcional donde aparecen los procesos más importantes de cada uno de los niveles (núcleo del sistema, de gestión, área de conocimiento y de soporte).

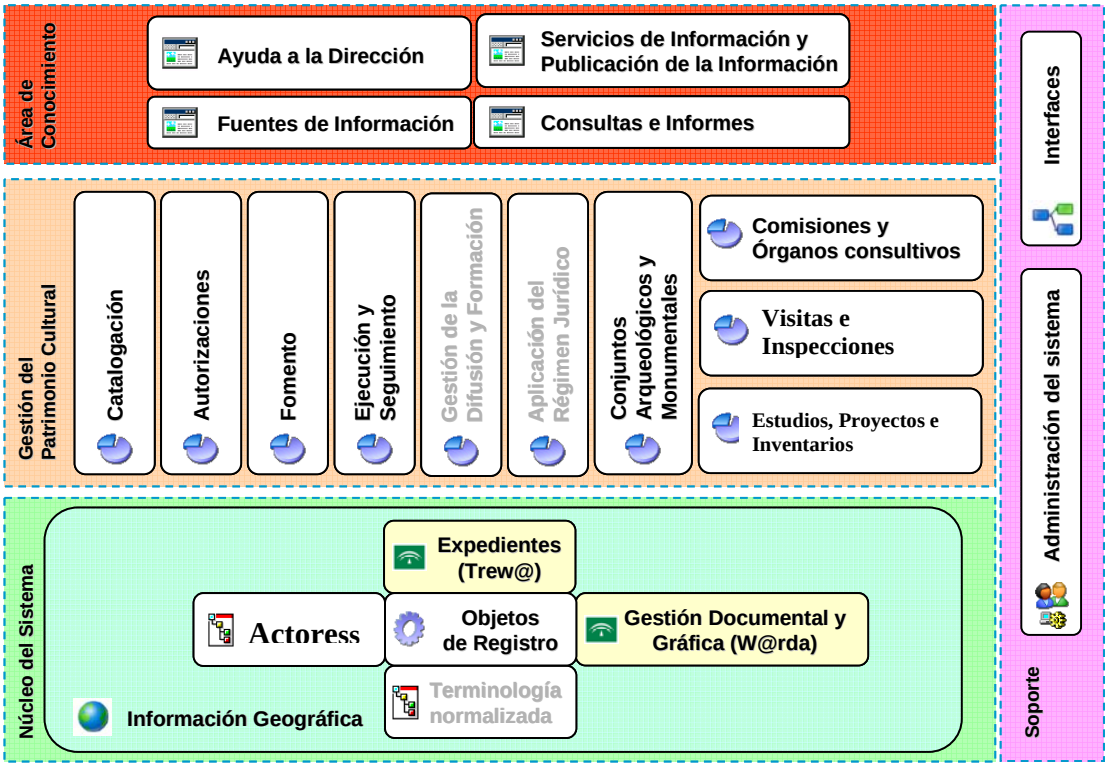

Figura 1- Arquitectura Funcional de Mosaico

3.2. Planificación, metodología y arquitectura

El proyecto que debía concluir con la construcción e implantación del sistema Mosaico, se planificó para ejecutar de forma secuencial 3 etapas, que a su vez tenían que ir dando lugar a distintas fases, a medida que el alcance se iba acotando. La primera fase fue el estudio de viabilidad, la segunda el análisis y diseño técnico con un alcance limitado a un conjunto de subsistemas y la tercera ha sido la construcción e implantación del sistema.

Centrando la planificación en esta tercera fase, la construcción se planificó para que tuviese tres entregas. La primera entrega sería la arquitectura del sistema, la segunda sería un subsistema construido enteramente sobre esa arquitectura y la tercera sería la del resto de subsistemas y documentación para la implantación.

La arquitectura del Mosaico (figura 2) desde un punto de vista físico ha sido construido como un sistema de n capas realizando una clara separación entre las capas de presentación, servicio y almacenamiento de datos.

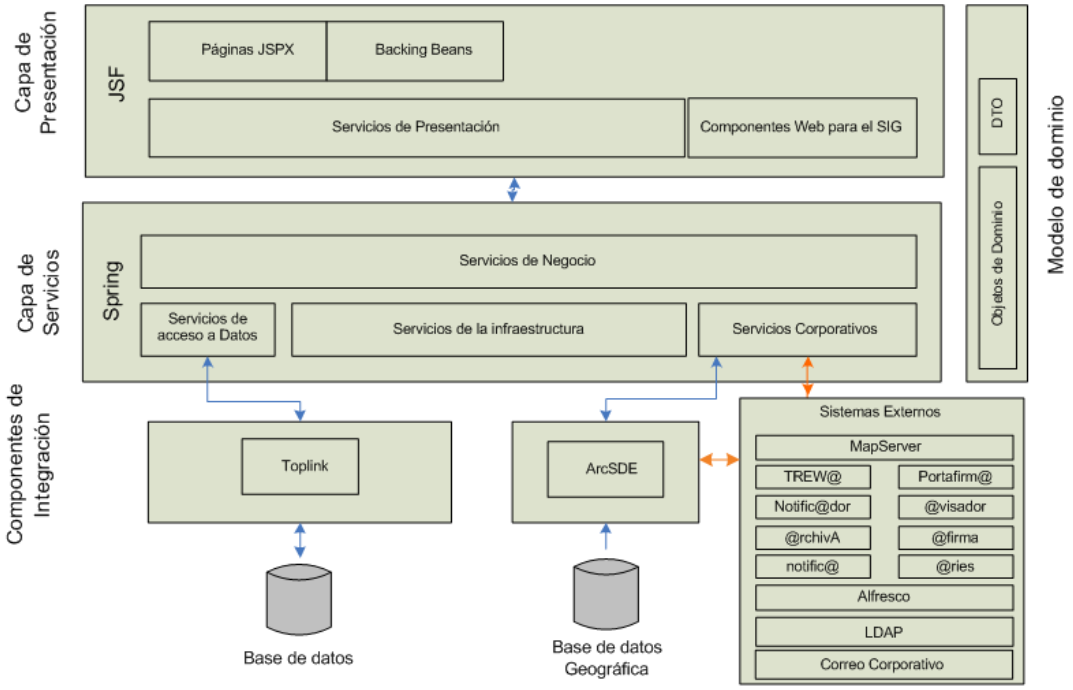

Figura 2. Arquitectura técnica de Mosaico

- Capa de Presentación: proporciona el contexto de presentación dentro de un navegador de Internet como Microsoft Internet Explorer o Mozilla Firefox que permiten acceder a los datos remotos a través de la Web. Para la implementación de esta capa se siguen las especificaciones de JSF 1.1, utilizándose las implementaciones Oracle ADF Faces y Apache MyFaces.
- Modelo de dominio: Entidades y objetos de transferencia de datos comunes entre las capas de servicios y presentación. Contiene las entidades de la aplicación. En la mayoría de los casos, se corresponden con una tabla del modelo de datos.

- Capa de Servicios: conjunto de componentes que resuelven la funcionalidad del sistema (Servicios de Negocio -contienen la lógica que trata con los objetos de dominio.-) o que proporcionan servicios especializados horizontales en el sistema de forma independiente a la funcionalidad (Servicios de la infraestructura para soporte de las funciones del propio sistema, Servicios Corporativos para el acceso a los sistemas corporativos de la Consejería de Cultura o a sistemas externos y Servicios de acceso a datos para acceder a los componentes de integración).
- Componentes de Integración: Para el acceso a la base de datos alfanumérica se ha utilizado el motor de persistencia Toplink; este componente resuelve el mapeo en el Modelo de Objetos y el Modelo Físico de Datos, encargándose de las conversiones de datos, gestión de concurrencias y transacciones. Para el acceso a la base de datos geográfica se utiliza el motor ArcSDE que permite el almacenamiento y consulta de información geográfica con una base de datos no geográfica.

3.3. Organización y coordinación en Mosaico.

La organización del proyecto de construcción e implementación de Mosaico, ha estado compuesta principalmente por las siguientes figuras:

- Director del proyecto. Tiene asignadas como responsabilidades, entre muchas otras, las de dirigir, supervisar y coordinar la realización y desarrollo de los trabajos así como aprobar el programa de realización de los trabajos.
- Equipo de Coordinación. Equipo multidisciplinar de técnicos que asesoran al Director del Proyecto y hacen de enlace entre usuarios y equipo de trabajo de la empresa.
- Jefe de Proyecto. Su responsabilidad, la ejecución de los trabajos.
- Equipo de trabajo. Responsable de la realización de todos los trabajos detallados.

Como soporte en el área de calidad del software y pruebas del sistema el proyecto ha contado con la ayuda de la Oficina Técnica de Calidad de la Consejería de Cultura.

4. Organización de las pruebas en Mosaico

Debido a la envergadura del aplicativo Mosaico, la organización y realización de las pruebas fue una fase compleja dentro del ciclo de vida. Había que partir de un entorno

similar al que finalmente se implantase en producción, con lo que la validación final del sistema discurriría por la validación individual de dos vertientes. Por un lado, la validación del aplicativo junto a su funcionalidad asociada y por otro la validación de la migración de los datos.

Para la primera, se prefirió comenzar con pruebas que determinaran la corrección de cada uno de los módulos que integraban la aplicación de forma separada para finalmente pasar a probar la integración entre diferentes módulos. En la primera versión entregada se probaron 19 subsistemas que iban desde Tesauro a módulos de Sistemas de Información Geográfica. Todas las pruebas que se llevaron a cabo fueron descritas sobre Enterprise Architect (www.sparx.com) y verificadas por el equipo de pruebas haciendo especial hincapié en los errores funcionales, así como los de validación de datos introducidos, rendimiento del sistema, etc.

Para la segunda, la validación de la migración de datos, hubo un equipo encargado del cotejo y aseguramiento de la fiabilidad de los datos migrados. Dicha validación fue un proceso bastante complejo de verificar tras el traspaso de los datos con los cuales se estaba trabajando actualmente en las diferentes aplicaciones a la migración de los mismos llevados a cabo sobre Mosaico.

4.1. Un perfil para la definición de las pruebas funcionales

Cuando un sistema engloba un alcance funcional tan grande como Mosaico, es necesario soportar su seguimiento y control mediante una herramienta case. Como se ha comentado, en este caso, la herramienta para esta gestión fue Enterprise Architect. Enterprise Architect es una herramienta case que permite el modelado de todos los elementos definidos en UML (Unified Model Language)[4]. Dentro de la Consejería se utiliza la metodología NDT y su conjunto de herramientas NDT-Suite (www.iwt2.org/ndt-suite.php). NDT-Suite incorpora varias herramientas basadas en la tecnología MDE (Model-Driven Engineering) [5] que dan soporte al desarrollo de aplicaciones mediante NDT.

La herramienta básica de esta suite es la denominada NDT-Suite. NDT es una metodología de desarrollo que soporta el ciclo de vida completo y que está sustentada sobre Enterprise Architect. NDT tiene definido un perfil de UML que es el que se ha implantado dentro de Enterprise Architect, lo que genera la herramienta NDT-Profile.

La ingeniería de requisitos de Mosaico estaba definido usando este perfil lo que ofrecía un repositorio estructurado de todos los requisitos. A la hora de plantear un sistema eficiente que gestionase las pruebas, se extendió dicho perfil incorporando en NDT-Profile un perfil específico para soportar las pruebas. De esta forma, se incluyeron modelos específicos para tratar con las pruebas unitarias, de integración, de sistema, funcionales y de implantación. Otra de las herramientas que incluye NDT-Suite, denominada NDT-Driver, permite ejecutar una serie de transformaciones que se encuentran definidas en la propia metodología NDT.

En la figura 3 se muestra un ejemplo de cómo el perfil ofrece una prueba funcional. La figura muestra la representación de un circuito funcional para la validación del alta de los objetos de registro (los elementos susceptibles de definirse como bien patrimonial).

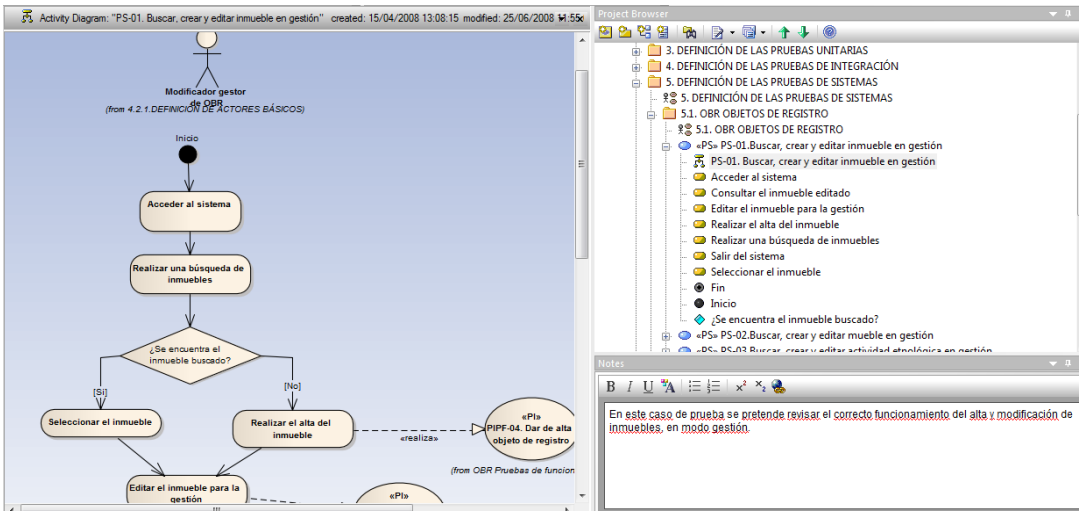

Figura 3. Estructura EA para el plan de pruebas

El procedimiento a desarrollar en cada una de las actividades viene descrito mediante escenarios en la pestaña correspondiente a la actividad tal y como se muestra en la figura 4.

Todas las pestañas que se observan en la figura, no hacen más que ser complementadas para que los elementos de nuestro diagrama de actividades que describe la prueba, estén los más definido, correcto y conciso posible. Una de las pestañas a destacar, sería ‘Links’ que son los enlaces que tiene dicho elemento con otros del diagrama pudiendo así recorrer de forma ordenada todo el árbol de elementos a probar.

Los casos de prueba también se enlazan mediante estos links con los requisitos funcionales pudiendo establecer qué cobertura exacta de las pruebas funcionales se está alcanzando.

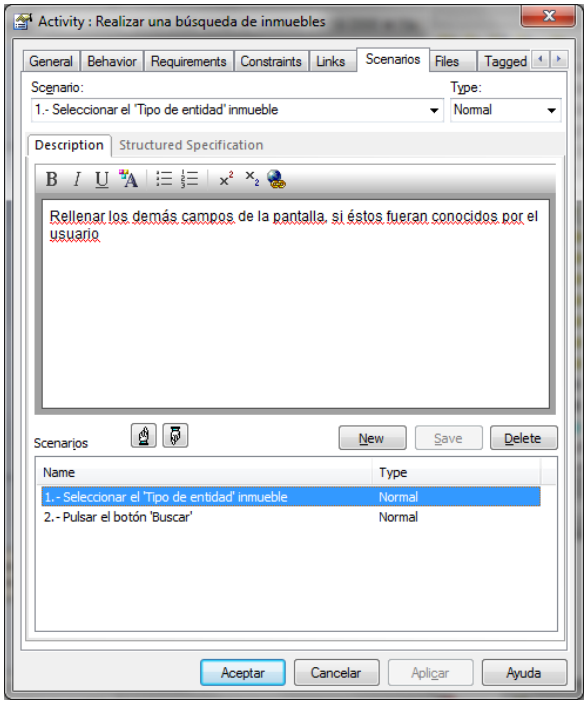

Figura 4. Escenarios de las propiedades de cada elemento

Además de la posibilidad mediante los links, para controlar de forma sistematizadas que cada una de las pruebas van cumpliendo cada uno de los requisitos marcados inicialmente, el *profile* lleva almacenadas una serie de matrices de trazabilidad que detectan cuando los objetivos que se marcaron en un inicio, se van cumpliendo. Dichas matrices trazan las diferentes relaciones entre objetos de nuestro proyecto, destacando la relación trazada entre los objetivos definidos para el sistema en base a actores, frases, nuevas naturalezas, prototipos de visualización y requisitos de almacenamiento. Del mismo modo se trazan varias matrices que recogen la información destinada a cubrir las necesidades de requisitos funcionales gestionadas por actores, así como tantos requisitos funcionales que terminaron creándose como prototipos de visualización. Dichas matrices, se iban completando automáticamente derivándose de la propia definición de pruebas diseñada sobre la propia herramienta anteriormente mencionada. En la figura 5 se pueden ver todas

las matrices de trazabilidad definidas previamente para la aplicación, donde incluso se trazaron a nivel de subsistema.

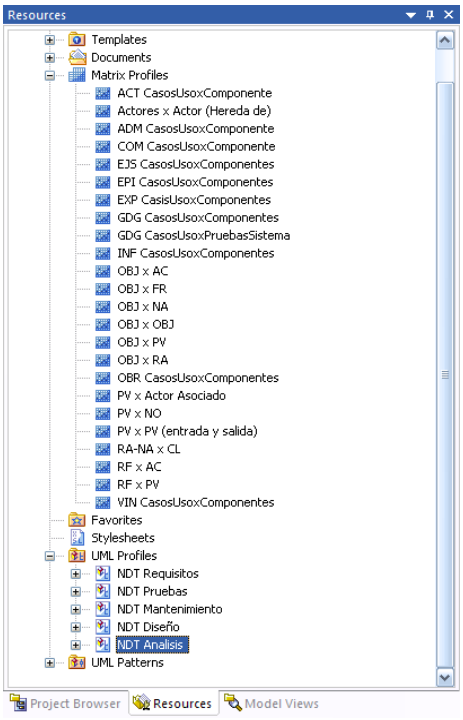

Figura 5. Diferentes matrices de trazabilidad

En otro orden, aplicando los programas adecuados como Selenium (http://seleniumhq.org/), se crearon un conjunto de pruebas iniciales y totalmente automatizadas, para pequeñas pruebas repetitivas, simplemente pulsando sobre algún botón, se realizaría de nuevo la prueba devolviendo el resultado de tal manera que rápidamente podríamos ver la verificación de la prueba, así como el lanzamiento automatizado de varias pruebas funcionales en forma de bloque. El funcionamiento de Selenium era tan simple que lo único que había que realizar la prueba una primera vez, usando Selenium a modo de grabadora de acciones sobre la aplicación Web. Dicho programa grababa cada acción del usuario dentro de la aplicación mediante la interfaz, qué valores incluir en qué combos, etc. Una vez grabada la prueba, siempre para pruebas determinadas como sencillas podíamos repetirla tantas veces como fuera necesario, cambiando valores de variables, etc. Pruebas tal y como la introducción de usuario y clave para ingresar a cada uno de los subsistemas para comprobar que el acceso se producía correctamente, al lugar indicado y que el usuario

con el cual accedía tenía concedido los permisos que debería tener, han sido ejemplos de pruebas que se han probado múltiples veces bajo la interfaz de Selenium.

4.2. Herramientas para la ejecución de las pruebas

Para la realización de la fase de pruebas, dependiendo del tipo de cada una de ellas se usó uno u otro programa de ayuda a la realización de dichas pruebas, aparte de la implicación de los diferentes grupos de trabajo por los cuales ha sido posible la realización e implantación de dicho software.

Para las pruebas unitarias y realizadas por el equipo de desarrollo, se usó JUnit (www.junit.org/) encargadas de verificar la corrección del código de forma modular. Dichas pruebas, se encargaron de verificar la correcta funcionalidad del código diseñado de manera individual, función por función, de manera exhaustiva. Para ellas, al contar con personal cualificado en el equipo de trabajo de desarrollo, los programadores no necesitaron formación alguna para llevar a cabo las pruebas así como el uso implícito de la herramienta en todas y cada una de ellas. Todas las pruebas estaban minuciosamente creadas para llevar a cabo un test de regresión agrupados por subsistema y funcionalidades donde simplemente con pulsar un botón en JUnit se llevaban a cabo todas las pruebas observándose rápidamente el resultado que podría haber ocasionado algún cambio en la estructura de la aplicación. Posteriormente, y realizadas por el mismo equipo de desarrollo, se llevaron a cabo las pruebas de integración encargadas de verificar la interconexión entre todas aquellas funciones mencionadas anteriormente y que son relacionadas unas con otras para abarcar la funcionalidad prevista durante la fase de análisis.

Para las pruebas de sistemas, aquellas que se encargan de probar la funcionalidad del sistema en su conjunto se dividieron por subsistema y se encargaron de llevarlas a cabo el equipo de pruebas siempre asesorados por la figura de los diferentes jefes de proyecto del sistema. La validación consistió primeramente en, por subsistema, recopilar todas y cada una de las pruebas definidas para dicho subsistema declaradas en la herramienta Enterprise Arquitect, tal y como definimos en el apartado anterior. También se realizó un completo y exhaustivo proceso de validación campo a campo, formulario a formulario, donde aquellas entradas que permitan sean las correspondientes a cada uno de ellos para así evitar los primeros errores que pudieran dar la conversión de datos no esperados en el envío de peticiones al servidor. Simplemente la idea era no dar nada por sentado y realizar las

pruebas desde el más absoluto "desconocimiento" de la semántica de la aplicación para intentar pulir todos aquellos defectos de validación de formularios que pudiera haber.

A continuación, e introduciéndonos ya en la semántica de la propia aplicación se fueron realizando las pruebas destinadas a validar todas y cada una de las funciones que el sistema entrañaba, pruebas de sistemas que precisaban de la utilización de funcionalidades de varios subsistemas. Dichas pruebas precisaban de un alto conocimiento semántico de la aplicación, siempre dirigidas y asesoradas por los diferentes jefes de proyecto que integraban parte del equipo de dirección. Tales pruebas, requerían de conocimiento para tramitar expedientes por medio de la aplicación sobre Trew@, interacción con diferentes plataformas como @ries, port@firma, etc. Pruebas que fueron complejas de realizar y validar debido a la alta casuística posible en cada uno de los escenarios de tramitación de los diferentes expedientes. Dichos escenarios descritos en las pruebas deberían siempre tener caminos congruentes a las soluciones o errores a validar en cada uno de ellos, no siendo posibles, siguiendo paso a paso el detalle exacto de la ejecución de la prueba llegar a un camino sin retorno en el sistema, o bien por falta de datos maestros (parametrización) en el sistema, o bien por mala ejecución de la misma, en definitiva, todo este tipo de excepciones o errores, deberían ser controlados por el programa para que en todo momento sea posible recuperarse del estado al que se llegue, mostrando por ejemplo mensajes de excepciones en los cuáles haya código o determinado texto que no debería poder ser visible por cualquier usuario.

Para todo este complejo proceso se unieron físicamente personal de desarrollo junto a sus jefes, personal de la Oficina Técnica de Calidad de la Consejería de Cultura así como un jefe de proyecto de la propia Consejería. Todos ellos con la misma dirección de trabajo, la de aunar las fuerzas para que el producto final tuviera el menor número de errores posibles a la hora de presentarlo al usuario final y como consecuencia el cliente. Para el control de todas las incidencias detectadas durante toda esta fase de pruebas, había varias herramientas de gestión de proyectos que nos facilitaban la labor de dar de alta dichas incidencias, para que posteriormente fueran tratadas y subsanadas por el equipo de desarrollo. Dichas herramientas eran entre otras, Mantis (www.mantisbt.org) y RedMine (www.redmine.org). La primera de ellas era la que servía de nexo de unión entre el equipo de desarrollo y la Oficina Técnica de Calidad y la segunda usada, tras la implantación del

proyecto durante su fase de mantenimiento donde se registraban aquellas incidencias detectadas durante el uso de la aplicación por los usuarios finales.

Para cada una de estas incidencias, había que debatir en consenso todas las partes implicadas, si llegaba a haber alguna duda de que dicha incidencia fuera correctiva o evolutiva (requerimientos no marcados al inicio del proyecto). Debido al retraso que había en la entrega del proyecto había requerimientos inicialmente válidos que con el paso del tiempo ya no lo eran, dando lugar en el futuro a muchas incidencias de este tipo denominadas como "indeterminadas" a la espera de debatirlas y darle una categorización adecuada, eran requisitos iniciales que tenían que ver con procedimientos que en su momento eran legales de una forma y que con el paso de los años habían sufrido algún ligero cambio en la Administración en su forma de proceder y hacía que el que se encontraba actualmente fuera antiguo, así como documentos formales propios de la misma que hayan cambiado su estructura, etc.

Sobre las pruebas de implantación, aquellas que comprueban el funcionamiento correcto del mismo en el entorno de operación, se llevaron a cabo por el equipo de implantación de sistemas siguiendo el plan establecido inicialmente para llevar a cabo, de forma correcta, la implantación del sistema.

Finalizando el proceso de pruebas, se llegó a las pruebas de usuario (User Test), pruebas importantes donde las haya donde se le muestra, a un grupo reducido de usuarios finales, un prototipo de la aplicación que finalmente tendrían para recoger primeras impresiones así como todo aquello que pudiera mejorar lo desarrollado hasta el momento. Se fueron planificando diferentes sesiones para cada uno de los subsistemas en los que se dividía Mosaico donde para cada una de ellas, fueron convocados miembros de todos los departamentos, personal técnico y formativo para mostrar, explicar y recoger impresiones de los usuarios finales que terminarían usando la aplicación. Dicha fase, fue un momento duro de trabajo donde la retroalimentación por parte de los usuarios hacia los formadores, hizo ir puliendo aún más, hacia la finalización correcta y satisfactoria de la aplicación. Cuando hablamos de momento duro nos referimos al hecho de recoger impresiones de usuarios finales, sus quejas hacia la aplicación y demás motivadas muchas de ellas por el retraso de la entrega y que la consecución de las pruebas no fuera lo esperado por ellos en algunos de los subsistemas entregados. Poca participación por este tipo de personas,

usuarios expertos, en la definición de funcionalidades así como en la elaboración del plan de pruebas podría ser una de las posibles causas de dicho rechazo. Los usuarios se acostumbran a lo que tienen y cuando algo funciona, al principio tienen reticencia a necesitar formación en una aplicación que a la larga, y si todo ha salido según lo previsto, le facilitará su labor diaria.

Al igual que los jefes de cada uno de los áreas discutían sobre alguna posible incidencia que apareciera durante la fase de pruebas de sistema o anterior, tras la revisión de todas las pruebas de aceptación por parte del usuario final, se elabora un documento que refleje todas las impresiones obtenidas de los propios usuarios, errores, etc. y de nuevo se vuelve a valorar la calidad del entregable distinguiendo de todo ello errores que pertenecieran al correctivo a entregar, mejoras o funcionalidades que el usuario no estuviera de acuerdo, observando para ello siempre los requerimientos solicitados inicialmente por el cliente, etc. Toda esta conclusión de cambios, errores, etc. finaliza en una nueva iteración de todas y cada una de dichas fases para probar todo aquello que hubiese sido susceptible de cambio así como probar algunos de los llamados fallos colaterales de la inclusión de lo nuevo en el proyecto.

Por último y otras de las pruebas importantes del proyecto era el rendimiento ofrecido por la aplicación ante consultas simples o complejas que implicaran refresco de pantallas con gran cantidad de datos. Inicialmente, como todo en general, los tiempos de respuesta ofrecidos fueron ligeramente altos para lo que realmente se solicitaba por parte de la dirección del proyecto, requiriendo en algunos casos la reformulación de la fase de análisis del proyecto de cara a mejorar la calidad ofrecida a nivel de tiempos de respuesta. Los tiempos, de forma general, fueron entrando poco a poco dentro de los baremos esperados desde coordinación para ser considerados como aceptables de cara a la futura implantación del sistema. Al precisar de un esquema de bases de datos tan complejo para albergar la gran cantidad de datos analizados y estudiados, redunda en tener gran cantidad de consultas con una alta densidad de datos a presentar en su devolución y con ello una tardanza implícita en la propia consulta difícil de subsanar de por sí, pero con los conocimientos necesarios para ellos y el refinamiento y redefinición de todos los componentes a nuestro alcance, se consigue finalmente minimizar el tiempo de respuesta.

Ligadas a estas pruebas, se sometió al sistema a determinadas pruebas de stress, aquellas encargadas de simular una carga de trabajo de n usuarios conectados a la vez realizando peticiones de datos al servidor y navegando de forma aleatoria por la aplicación. Para realizar estas pruebas se utilizó otra de las aplicaciones propias del mercado como es JMeter (http://jakarta.apache.org/jmeter/). Se simulaban conexiones y peticiones al servidor de consultas de las más pesadas correspondientes a los subsistemas de más almacenamiento de carga, llegando a simular alrededor de 50 usuarios realizando dichas peticiones con unos tiempos de respuesta aceptables en todas y cada una de ellas.

Algo más de tres meses de tiempo fue destinado a la fase de pruebas, quizás poco tiempo debido a la envergadura del proyecto y mucho más del previsto inicialmente para las mismas. Un proyecto de tal magnitud, quizás necesitara de mayor tiempo puesto que había mucha interacción con otras aplicaciones de Administración que ya de por sí ofrecían problemas que siendo ajenos al desarrollo del proyecto, no dejaban de ser una fuente de problemas con el usuario. Fue en la primera revisión de pruebas de sistemas que se realizó, donde los resultados que se produjeron, hicieron finalmente verse incrementados los tiempos para las pruebas y un mayor foco de atención a las mismas.

4.3. Resultados

Tras la realización de las pruebas de sistema, implicó una serie de iteraciones en el proceso de actualización de errores detectados, de mayor y menor gravedad, que implicó la repetición en cada una de ellas de todas y cada una de las pruebas en las cuales consistía la validación del sistema. La persistencia en la precisa y correcta validación de dichas pruebas hizo posible, de carácter general, del buen resultado final del sistema.

La primera de las entregas resultó tener diversos fallos de diverso tipo y diversa consideración. Aparecieron los primeros problemas de integración entre los subsistemas así como errores de validación, excepciones no controladas y demás problemas, incidencias, incongruencias de datos, etc.

Hacer constar la gran importancia que supone el desarrollo de las pruebas de aceptación con el usuario final, reuniones mantenidas entre jefes de proyecto y colaboradores de diferentes ámbitos y departamentos junto a una muestra relevante de personas del grupo final de usuarios que terminarán accediendo a las diferentes aplicaciones. Pruebas en las que el usuario final, ve por primera vez plasmado sobre una

aplicación, los requisitos y objetivos a cubrir por dicha aplicación, recogiendo de éstos, sus impresiones, así como cualquier detalle a pulir en la misma que haga que la entrega final sea, todo lo satisfactorio posible para todas las partes implicadas en el desarrollo de tan complejo software.

De las últimas revisiones que se detallaron se encontraron 77 incidencias de diferente índole y consideración tales y como se muestra en el gráfico siguiente.

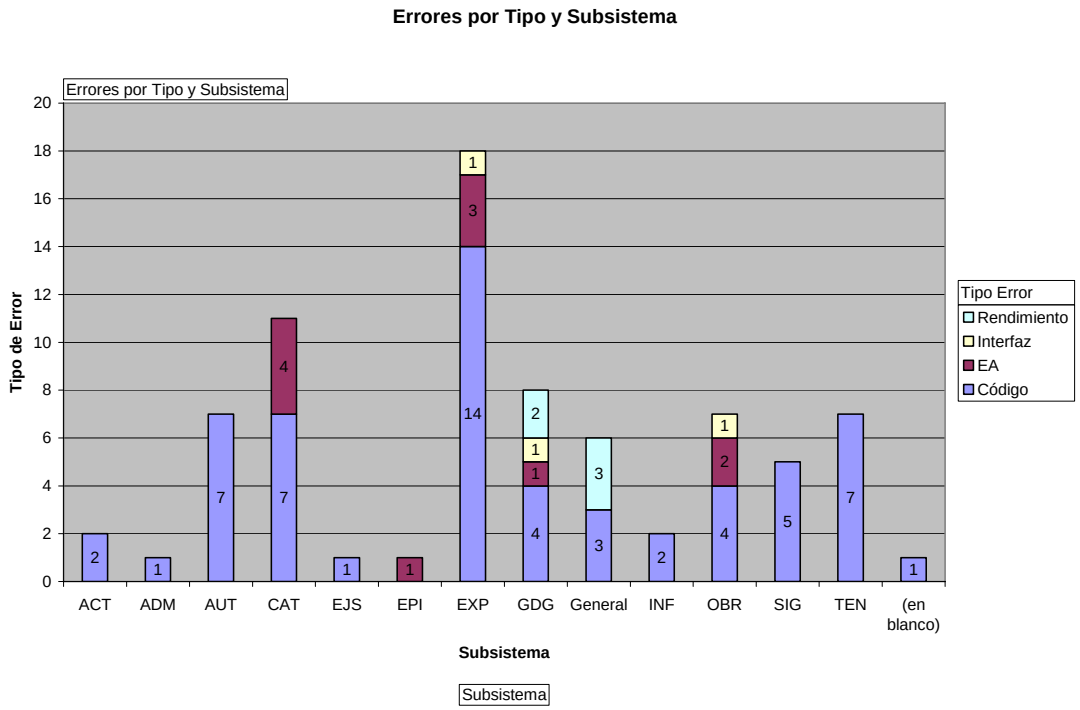

Figura 6. Errores de versión por subsistema y tipo

Cuando hablamos de errores de tipo EA, nos referimos a errores de documentación donde la desactualización de una prueba en base a la nueva funcionalidad deja patente el fallo que habría que subsanar también, dando lugar a equívocos en el buen desarrollo de la prueba.

5. Conclusiones y trabajos futuros

Este artículo ha presentado un ejemplo práctico de la realización de la fase de pruebas en un proyecto de envergadura denominado Mosaico. Llegados a este punto, se puede concluir que el trabajo de ejecución y organización de las pruebas en un sistema de información no

solo afecta a la parte técnica si no que es necesario establecer una serie de políticas de trabajo y organización que garanticen los resultados.

El artículo ha presentado el entorno del proyecto, ha descrito el entorno tanto de organización como de herramientas usados para el proyecto y ha presentado las decisiones organizativas y estratégicas que hubo que establecer.

El proyecto, además, potenció la incorporación de las pruebas dentro de las fases cubiertas por NDT y tras esta aplicación, todas las herramientas de NDT-Suite se han enriquecido para soportar también esta fase.

En la actualidad, no solo los proyectos que se ejecutan en la Consejería utilizan este entorno, si no que en otros proyectos en el entorno empresarial están usando el profile definido para las pruebas.

Desde su uso en Mosaico, se ha mejorado en el ámbito de las pruebas de de sistema y su descripción en Enterprise Architect, haciendo que los casos de prueba sean más lineales y separando diferentes alternativas de ejecución en pruebas diferentes. También se ha mejorado en el proceso de pruebas de aceptación y estas ya no son ejecutadas sólo por los usuarios sino que tienen el soporte del área TI. También se ha visto la idoneidad de tener un entorno de pruebas de usuario diferente del de preproducción, para permitir la ejecución de las pruebas y permitir un periodo de adaptación de los usuarios a las novedades del sistema.

En un futuro inmediato se pretende tener un conjunto de pruebas unitarias automatizadas, que permitan su ejecución antes de cualquier nuevo despliegue que permita verificar que los módulos básicos siguen funcionando según lo esperado, y profundizar en la trazabilidad desde los requisitos funcionales, traducidos en casos de uso, y las pruebas de sistema que se describen en los planes de pruebas.

Además se están mejorando las transformaciones para la generación sistemática de las pruebas dentro del entorno.

Referencias

[1] Jalote, P., *Software Project Management in Practice*, Addison Wesley, 2002.

[2] Consejería de Cultura, Junat de Andalucía, “Sistema de Gestión del Patrimonio Cultural: Mosaico”, Disponible en:

www.juntadeandalucia.es/cultura/web/areas/bbcc/sites/consejeria/areas/bbcc/sistema_gestion_bienes_culturales (visitado octubre de 2010)

[3] Escalona M.J.y Aragón, G., "NDT: A Model-Driven Approach for Web requirements", *IEEE Transactions on Software Engineering*, vol. 34, nº 3, 2008, pp. 370-390.

[4] OMG, *Unified Modeling Language 2.1.1 Superstructure Specification (OMG doc. formal/07-02-05)*, OMG, 2007.

[5] Moreno, N., Romero, J.R. y Vallecillo, A., "An Overview of Model-Driven Web Engineering and the MDA". En: G. Rossi, O. Pastor, D. Schwabe, L. Olsina (eds.), *Web Engineering: Modelling and Implementing Web Applications*. Springer-Verlag, 2007.