UNIT4 – SCRUM: Gestión de distribuciones

Francisco Cabello Lara

Resumen—El desarrollo y el mantenimiento correctivo y evolutivo de los proyectos informáticos, necesita de una buena gestión de los componentes modificados. Actualmente, en UNIT4, los equipos de R&D utilizan métodos diferentes para llevar el control de los componentes modificados para cada petición de cambio (CR) y esta circunstancia dificulta su gestión. El objetivo de este proyecto es incorporar un módulo nuevo en la aplicación ekon.SCRUM de UNIT4 que permita registrar los componentes modificados para cada CR resuelta y automatizar la documentación y las distribuciones realizadas. Se ha realizado un estudio de las diferentes herramientas de mercado, pero finalmente se ha decidido integrar la solución en la propia herramienta de la empresa para el desarrollo del proyecto. Desde el esquema teórico se ha desarrollado la aplicación contemplando todas las necesidades de cada equipo de R&D. Se ha completado satisfactoriamente el proyecto dentro del calendario previsto, ofreciendo un producto que aporta una serie de beneficios en el funcionamiento diario de UNIT4.

Palabras clave—karat, Componentes, Petición de cambio (CR), Kit de distribución, FIX, Investigación & Desarrollo (R&D), SCRUM, Pruebas cruzadas, SDIC.

Abstract—A good management of the modified components in the corrective and evolutionary development of products in the company UNIT4, is key issue for its operation. Currently each R&D team uses a different way to keep track of the changed components for each change request (CR) and it difficults its management. The goal of this project is to incorporate a new module in the application of UNIT4 ekon.SCRUM allowing to record the modified components for each CR and to manage the distributions that are made. We performed a study of the various market tools, but finally the solution involves using the proprietary tool of the company to develop the project. From the theoretical framework the application has been developed considering all the needs of the R&D team. The project has been successfully completed in schedule, offering a product that provides a number of benefits in the daily operation of UNIT4.

Index Terms—karat, Components, Change Request (CR), Distribution kit, FIX, Reserch & Development (R&D), SCRUM, Cross tests, SDIC.

1 INTRODUCCIÓN

REALIZAR una aplicación para el registro de componentes puede ser un trabajo complicado y muy laborioso si no se dispone de las herramientas adecuadas. En el pasado las herramientas eran básicamente hojas de cálculo y editores de texto, que permitían introducir de manera manual los componentes que se hubieran gestionado, que en general estaban enfocadas más a la documentación que al control y gestión de las distribuciones. En los últimos años se ha desarrollado una nueva herramienta, karat [2]. Se trata de una herramienta que permiten crear y mantener aplicaciones con facilidad, encargándose de los trabajos más pesados y que ocupaban gran parte del tiempo.

El presente proyecto se ha realizado en la empresa UNIT4. Se trata de una empresa de software que crea, desarrolla, comercializa y ofrece una amplia gama de productos de software de gestión y servicios relacionados que permiten que los clientes puedan asumir los cambios más radicales. Sus soluciones ofrecen un apoyo funcional de negocio tanto genérico como especializado, y benefician a todos los tipos de organizaciones del sector público y privado en todo el mundo. Además, algunas de ellas están específicamente centradas en las necesidades de sectores de mercado concretos. Cuenta con 4.382 empleados y con oficinas y distribuidores en diversos paises para garantizar un acceso fácil y local a las ventas, servicios y soporte.

Teniendo en cuenta este caso de UNIT4, cada equipo de R&D utiliza métodos distintos para gestionar los componentes modificados para cada control de cambio, y la generación de kits.

Hay además una necesidad que puede ser cubierta con la nueva herramienta. Se trata de la detección de dependencias entre diferentes kits de distribución, sobre todo cuando se trata de un adelanto urgente (FIX). Estas dependencias, hasta el momento se controlaban de forma manual, pudiendo generar la aparición de errores por omitir en la distribución un elemento dependiente de otro.

A partir de la situación descrita, se presenta este proyecto que pretende dar cumplimiento a todos los requisitos esbozados para permitir la obtención de una solución robusta al problema presentado, lo que garantiza la unificación en una sola aplicación de todas las necesidades de los equipos de R&D y la automatización del proceso.

Este artículo se iniciará con una breve descripción del estado actual del arte en relación a las funcionalidades de control de distribuciones. A continuación, se describen las

consideraciones preliminares tomadas, así como el establecimiento de los objetivos que deben cumplirse a fin de evaluar el éxito de este proyecto. Luego nos trasladaremos en más detalles técnicos acerca de la aplicación real de nuestra solución, explicando cada paso realizado. Finalmente concluiremos este documento con una sección de conclusiones con los objetivos logrados con éxito, las futuras extensiones y los agradecimientos.

2 ESTADO DEL ARTE

Actualmente hay algunos productos en el mercado que pretenden ofrecer unos niveles satisfactorios de servicios de registro de componentes, incidencias y gestión de distribuciones.

Uno ejemplo de estos productos es Prodware Desk de Microsoft [3], el cual dispone de funcionalidades de gestión de incidencias, gestión de problemas, gestión de cambios, gestión de activos, gestión de contratos, reporting y estadísticas, base de datos de conocimiento, entre muchas otras más.

Otro de los productos podría ser KMkey [4], indicado para servicios de mantenimiento, ayuda al usuario y resolución de problemas en cualquier sector. Permite definir flujos de trabajo para abordar problemáticas derivadas de anomalías en servicios y maquinaria. También permite gestionar la calidad y gestionar proyectos.

El problema de estos productos es que no son rentables ya que no se adecuan ni cubren con todas las necesidades de UNIT4, en cuanto a la automatización del registro de componentes ni de la documentación interna que genera de la empresa.

Nuestra aplicación pretende ser un servicio que cubra con todas las necesidades, garantizando una buena gestión de componentes modificados y nuevas distribuciones, además de efectuar su desarrollo bajo la herramienta karat con la cual trabaja la empresa de UNIT4.

2.1 Conceptos básicos

El desarrollo en R&D de UNIT4 se gestiona mediante **CR** que es la unidad mínima de desarrollo, tanto de los nuevos desarrollos como para el mantenimiento legal y correctivo.

Cada CR (por el inglés *Change Request)* resuelta tiene unos **componentes** que se controlan para poderlos distribuir.

Un **kit de distribución** incluye un número determinado de CR con todos sus componentes así como las directivas para la instalación.

El **mantenimiento legal, correctivo** o **adaptativo** se trata de las modificaciones de los productos teniendo en cuenta las leyes cambiantes del estado, por correcciones debidas a fallos, o adaptaciones a las nuevas tecnologías.

Las **pruebas cruzadas** son aquellas pruebas que realiza un técnico sobre los mantenimientos realizados, y que una vez efectuadas, vuelven a ser probadas por otro técnico distinto.

Cuando nos referimos a **SDIC**, nos referimos al repositorio donde se encuentra la información de los componentes registrada.

Un **release notes** es un documento que se distribuye junto al producto de software, frecuentemente cuando el producto aún está en fase de desarrollo o en estado de pruebas. Para los productos que están en uso por los clientes, el release notes es un documento suplementario que se entrega al cliente cada vez que un error se corrige o se hace una mejora en el producto.

En cuanto al **DIS0** es un documento que autoriza la publicación de la distribución.

Las **órdenes de archivo** hacen referencia a las solicitudes para que los diversos documentos que forman parte de las distribuciones queden guardados y registrados en el archivo informático de la empresa.

Una **evidencia** forma parte de la documentación solicitada por los organismos que auditan la actividad de la empresa, los cuales necesitan una constancia documental de los trabajos realizados.

3 OBJETIVOS

3.1 Objetivo principal

El objetivo principal del proyecto es la automatización de la gestión de componentes. Esto se resolverá con un software que, a partir de una CR se registren los componentes modificados y resueltos permitiendo identificar los componentes por versión e identificar las dependencias de manera automática.

Más concretamente, el proyecto plantea los siguientes objetivos técnicos y personales.

3.2 Objetivos técnicos

Este objetivo principal podemos dividirlo en una serie de objetivos más técnicos, es decir requisitos funcionales, que marcan la pauta a la hora de realizar el desarrollo del proyecto en función de su importancia. Estos objetivos son:

- Facilitar el registro de componentes por CR resuelta.
- Incluir la posibilidad de diferenciar los componentes según la versión.
- Generación automática de plantillas para distribuciones de versiones.
- Contemplar las dependencias entre CRs para facilitar la generación de kits de distribución.
- Mejora de la calidad de las distribuciones.
- Cierre de versiones.
- Generación automática de documentación en HTML con componentes modificados por versión (listados impresos).
- Generación automática de Release notes.
- Generación automática de la documentación de una distribución (dis0, órdenes de archivo).

- *E-mail de contacte: fco.cabello.lara@gmail.com*
- *Menció realitzada: Enginyeria del Software.*
- *Treball tutoritzat per: Josep Lladós (Centre de Visió per Computador)*
- *Curs 2013/14*

- Generación de las 'pruebas cruzadas'.
- Generación automática de la documentación de una prueba cruzada (evidencias).
- Creación de las plantillas HTML para la generación automática de las documentaciones (dis0, órdenes de archivo, evidencias).

En la Figura1 podemos observar la relevancia de los objetivos. Todos aquellos que son marcados como "secundarios" son objetivos que fueron previstos para la ampliación del proyecto.

Objetivos	*Críticos*	*Prioritarios*	*Secundarios*
Aprendizaje de metodología para desarrollo de software y aplicaciones de desarrollo e investigación.	X		
Facilitar el registro de componentes por CR resuelta.	X		
Incluir la posibilidad de indicar diferentes componentes por versión.		X	
Generación automática de plantillas para distribuciones de versiones.		X	
Búsqueda de dependencias entre CRs para facilitar la generación de FIXES.	X		
Generación de plantillas para realizar pruebas cruzadas.			X
Generación automática de documentación en html con componentes modificados por versión.			X
Generación de Release Notes.			X
Generación automática de órdenes de archivo al cerrar una versión.			X

Figura1. Relevancia de los objetivos.

3.3 Objetivos personales

A nivel personal me marqué una serie de objetivos a en la realización del proyecto y durante mi estancia en UNIT4 son:

- Adquirir experiencia laboral en una gran empresa del sector informático.
- Aprendizaje de metodologías para el desarrollo de software (SCRUM) y aplicaciones de desarrollo e investigación (karat).
- Mejorar el nivel de programación, análisis y desarrollo.

4 METODOLOGÍA

Se ha seguido la metodología de desarrollo ágil SCRUM [1] caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento implícito de las personas en equipos autoorganizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o de cascada.

SCRUM es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales son el de *ScrumMaster*, que mantiene los procesos y trabaja de forma similar al director de proyecto, el *ProductOwner*, que representa a los *stakeholders* (interesados externos o internos), y el *Team* que incluye a los desarrolladores.

Durante cada sprint, un periodo entre una y cuatro semanas (la magnitud es definida por el equipo), el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del *Product Backlog*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de *Sprint Planning*. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint.1 Durante el sprint, nadie puede cambiar el *Sprint Backlog*, lo que significa que los requisitos están congelados durante el sprint.

En nuestro caso se han descompuesto los objetivos del proyecto en historias, que son los requerimientos detallados del proyecto.

A cada historia se les dio una prioridad en función de la importancia que tenía en el proyecto y las posibles dependencias entre diferentes historias. Y las historias a su vez se descomponían en tareas a realizar.

A los *sprints* marcados (iteraciones) en la planificación se asociaron las historias a realizar durante ese periodo de tiempo. Normalmente los sprints oscilaron entre las 2 y 4 semanas en función de la dificultad y el trabajo previsto.

Una vez terminado el sprint se presentan los requisitos completados y se hacía un análisis de como se ha trabajado y los problemas surgidos para así poder mejorar la productividad en los siguientes sprints.

Gracias a esta metodología todos los cambios y nuevas necesidades surgidas durante el transcurso del proyecto, han permitido su realización sin penalización alguna en cuanto a tiempo y recursos.

La Figura 2 muestra el esquema de un proyecto desarrollado con la metodología SCRUM.

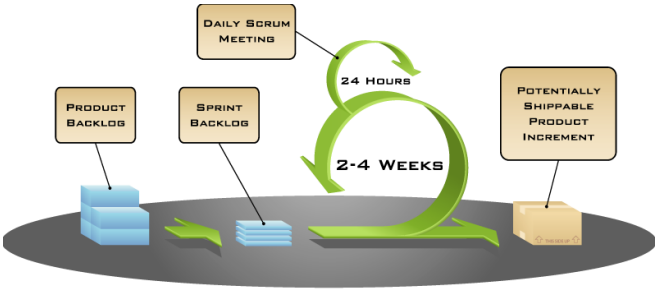

Figura2. Metodología de desarrollo SCRUM.

5 PLANIFICACIÓN

El proyecto se ha desarrollado en el transcurso de 19 semanas desde su inicio en Febrero de 2014 hasta Junio de 2014.

En el transcurso de la primera semana, se realizó una formación de la herramienta karat y la metodología de programación.

A partir de aquí el proyecto se dividió en 5 sprints, con una duración aproximada de entre 2 y 4 semanas cada uno, para el desarrollo del proyecto.

Las tres últimas semanas, comprendidas en el último sprint, se reservaron para la implantación del proyecto desarrollado en la aplicación de ekon.SCRUM de UNIT4 y las pruebas pertinentes, y a la realización de la documentación necesaria para terminar el TFG.

En la Figura 3 podemos observar del diagrama de Gantt que representa esta planificación del proyecto. En la cual no ha habido incidencias ni desviaciones, y los cambios y nuevos requerimientos introducidos a lo largo del proyecto no han afectado a la planificación inicial planificada gracias a la metodología utilizada.

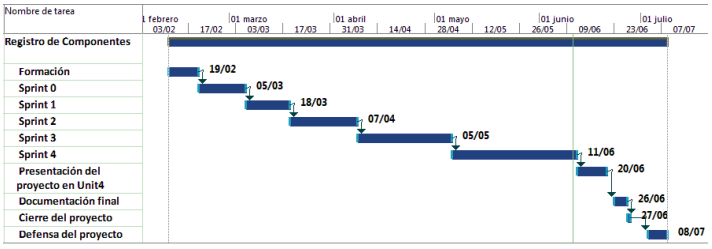

Figura 3. Planificación del TFG estructurada en Sprints.

6 HERRAMIENTAS UTILIZADAS

6.1 karat

El entorno de desarrollo sobre el cual se ha desarrollado todo el proyecto es **karat** versión 9.1 [6].

karat 9.1 es la infraestructura tecnológica de última generación de las soluciones ekon (software de gestión sin fecha de caducidad y con el menor coste de propiedad gracias a su exclusiva arquitectura karat, que permite disponer con rapidez y a muy bajo coste de las últimas actualizaciones y versiones que incorporan los cambios tecnológicos y funcionales que exige la competitividad empresarial.) desarrolladas por UNIT4. Totalmente desarrollado en el lenguaje de programación Java™, es independiente de los clientes de IT (Linux, Unix, Apple, Windows, etc.), no requiere de un explorador para trabajar en la web, se puede utilizar en cualquier dispositivo (PC, PDA, teléfonos móviles, etc.) y es compatible con todos los tipos de lenguas y alfabetos.

Otras características del sistema:

- Servidores web. Uso de servidores de aplicaciones (J2EE) o aquellos con soporte para servlets (versión 2.4).
- Ecosistema: Web Services, SOA, SaaS, EJB, XML, etc.
- Explotación de los formatos de documentos "open", usando el estándar ODF, Open Document Format.
- Funcionalidad multilingüe, datos multilingües, Unicode, etc.

karat garantiza la máxima independencia respecto a los elementos tecnológicos (bases de datos, sistemas operativos, terminales, etc.) y facilita un alto grado de autonomía en la automatización de los procesos de negocio, permitiendo que los cambios se produzcan de forma rápida, instantánea y casi sin coste. Además, permiten que las actualizaciones se realicen de forma inmediata.

Figura4. Herramienta de desarrollo de UNIT4.

6.2 Java

Java [7] es un lenguaje de programación desarrollado por Sun Microsystems de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

El principal motivo para escoger Java es que karat está desarrollado completamente con este lenguaje por su potencia y por su independencia de plataforma.

Figura 5. Plataforma Java.

6.3 Eclipse

Eclipse [8] es un Entorno Integrado de Desarrollo (IDE) extensible, compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar. Pese a que eclipse esté escrito en su mayor parte en Java (salvo el núcleo), se ejecuta sobre una máquina virtual de ésta y su uso más popular sea como un IDE para Java, eclipse es neutral y adaptable a cualquier tipo de lenguaje.

Debido a que karat se integra en Eclipse para el desarrollo y diseño, se ha elegido este programa de desarrollo.

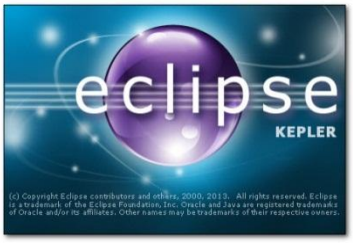

Figura6. IDE Eclipse Kepler.

6.4 SQL SQuirreL

SQL SQuirreL [9] es una herramienta de administración de base de datos. Utiliza JDBC para permitir a los usuarios explorar e interactuar con las bases de. Proporciona un editor que ofrece auto-completar de código y resaltado de sintaxis para el estándar SQL. Es software libre y se distribuye bajo la Licencia Pública General GNU.

Se ha elegido esta herramienta por previo conocimiento, para la comprobación de las consultas a la BBDD del proyecto.

7 ARQUITECTURA E IMPLEMENTACIÓN

Toda aplicación karat se caracteriza por tener una arquitectura concreta. Se trata de una serie de elementos independientes pero entrelazados entre sí y que le dan la versatilidad y flexibilidad deseada.

En la Figura 7 podemos observar el proceso y estructura que se sigue a la hora de implementar los mantenimientos y codificar el funcionamiento de la aplicación.

Figura7. Estructura de karat.

Lo primero que se crean son las tablas de la base de datos. Esto se hace a través del Eclipse, ya que karat se integra como una extensión de este al instalarse. Desde aquí definimos las tablas con los diferentes campos de cada una, el tipo y los enlaces entre tablas.

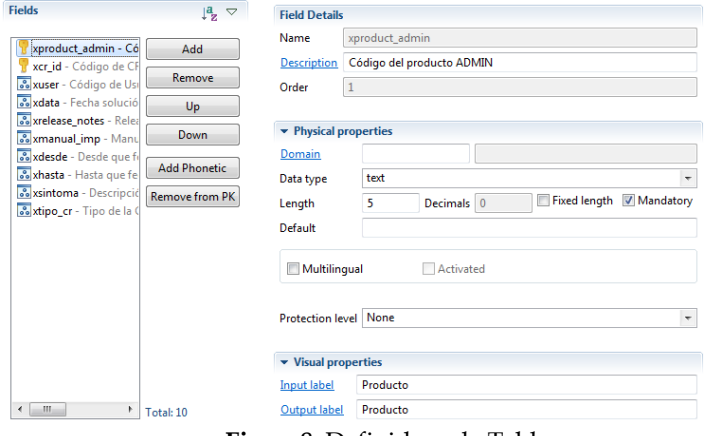

Figura8. Definidora de Tablas.

Una vez tenemos las tablas, el resto de procesos antes de llegar a la codificación se hacen a través de la diseñadora de karat (karat.studio), donde se definen las consultas base (Base Query) y las consultas (Query). Estas se generan de manera automática, como podemos ver en la Figura 9, con solo decirle a que tabla hace referencia y que campos queremos que aparezcan en las consultas.

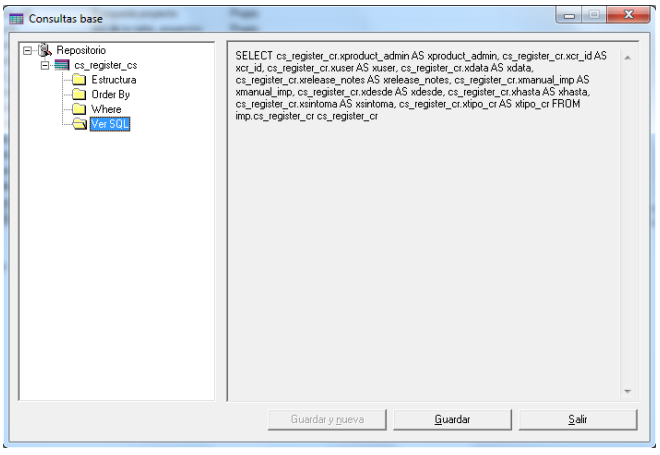

Figura9. Creación de las Consultas.

En los objetos de negocio se asocian todas las tablas entre ellas de un mismo mantenimiento, y se indican si hay campos calculados a través de consultas dependientes de otras tablas. Esto facilita mucho la labor a la hora de desarrollar, como podemos observar en la Figura 10.

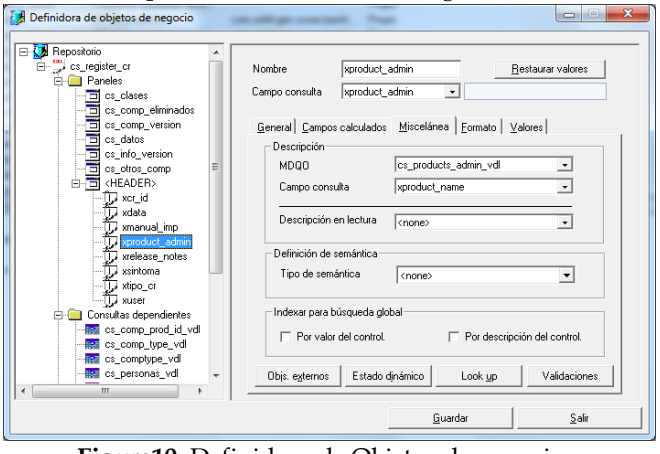

Figura10. Definidora de Objetos de negocio.

En cuanto a los formularios, es el diseño gráfico de cómo se va a mostrar al cliente toda la información a través de la aplicación de UNIT4. Utilizando los objetos de negocio previamente definidos y con los que posteriormente daremos funcionalidad codificándolos, como se muestra en la Figura 11.

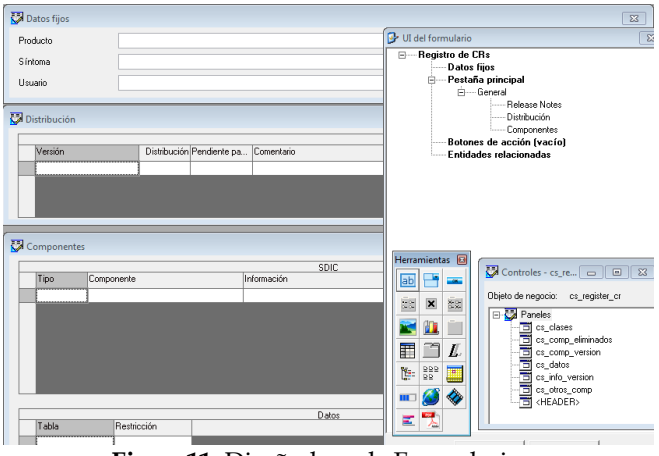

Figura11. Diseñadora de Formularios.

Los listados es un proceso muy parecido al de los formularios, pero el diseño de la información que se muestra se hace teniendo en cuenta que será mostrada a través de documentos impresos en diferentes formatos, por ejemplo .pdf, .docx, .xlsx, .html, etc. En la Figura 12 vemos la estructura para diseñar los listados.

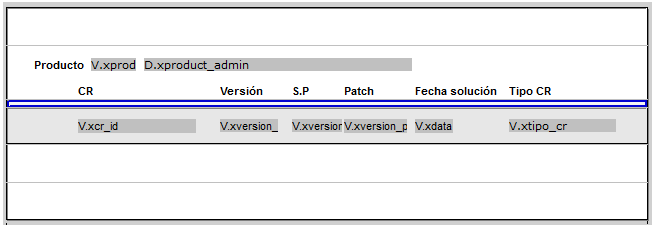

Figura12. Diseñadora de Listados.

Para realizar el proceso de codificación se vuelve al entorno de desarrollo Eclipse, donde se generan los scripts en función de los objetos de negocio y/o los formularios definidos anteriormente. Se realiza en Java pero utilizando las librerías, métodos definidos y las APIS propias de karat.

8 DISCUSIÓN DE LOS RESULTADOS

A continuación se ilustran los resultados de las diferentes funcionalidades implementadas en relación a los objetivos planteados del proyecto.

8.1 Facilitar el registro de componentes por CR resuelta.

En el mantenimiento de versiones de producto se especifica el producto y la distribución (versión) a la que hace referencia. Un mismo producto puede tener diferentes distribuciones, por tanto habrá que realizar tantas versiones de producto como distribuciones tenga.

Se realiza una carga de los componentes que se quieren registrar vía repositorio indicando sobre que configuración queremos hacer la carga, o a través de un documento tipo Excel.

8.2 Incluir la posibilidad de indicar diferentes componentes por versión.

En el mantenimiento del registro de componentes se especifica sobre que producto se quiere realizar la distribución y se indica la CR asociada. Automáticamente se informará del síntoma (breve descripción del problema) y se registrarán los componentes en función de las diferentes versiones indicadas en el mantenimiento de versión de producto.

Se podrán gestionar cuantas distribuciones tenga ese producto y siempre y cuando no esté cerrada esa distribución.

8.3 Búsqueda de dependencias entre CRs para facilitar la generación de FIXES.

Para un producto indicado aparecen todas las distribuciones (siempre y cuando no estén cerradas) y se puede seleccionar las que se quieran gestionar para generar el FIX. Al seleccionar las distribuciones aparecerán las CRs de cada distribución y se marca y se indican las dependencias entre ellas si las hay de manera automática y se indicará de quien es dependiente, además de aparecer los componentes registrados y sus dependencias en caso de haberlas.

8.4 Generación automática de plantillas para distribuciones de versiones.

Como se puede observar en la Figura 14 del anexo, en los componentes del "SDIC" y "Datos" tenemos unos botones para generar ficheros, que serán las plantillas para realizar un export de los FIXES.

El botón de "Seleccionar directorio" es para indicar en que directorio se va a guardar la plantilla para el FIX. En cambio, el botón "Generar fichero" genera un ".tet".

8.5 Generación automática de documentación en listados impresos.

Aquí es donde se realiza toda la explotación de la información según las necesidades establecidas por cada equipo. Hay diferentes listados:

- Listado de CRs por producto/versión con o sin desglose de componentes.
- Listado de CRs por fecha de solución con o sin desglose de componentes.
- Listados diversos de la "Generación de plantillas".

Al pulsar el botón de "Imprimir" nos sale un asistente para seleccionar en que formato queremos la información y por donde queremos hacer la impresión del listado (por pantalla o mandar a la impresora).

8.6 Generación de plantillas para realizar pruebas cruzadas.

Las pruebas cruzadas también van en referencia al producto y a una versión (pueden haber diversas versiones para un mismo producto). Está dividida en dos parrillas.

La primera nos índica de manera automática las CRs asociadas a ese producto y versión y hace referencia a la información final la cual se alimentará de la segunda parrilla y debería quedar con todas las pruebas pasadas correctamente.

La segunda parrilla son las diferentes versiones de una misma prueba, ya que puede dar el caso que se haya modificado alguna CR con posterioridad y se tenga que volver a pasar otra prueba para esa CR. Las CRs son "hyperlinks" que nos llevan al mantenimiento de "Registro de componentes" para ver de qué trata la CR (síntoma).

Se ha añadido como funcionalidad un botón para la generación de las "Evidencias", que es un documento en formato HTML donde aparecen documentadas todas estas pruebas.

8.7 Generación de Release notes.

La generación de Release notes se divide en diferentes pestañas que corresponden a los diferentes apartados del documento final.

Se genera por producto y versión y se indica de manera automática donde se encuentra la plantilla y donde queremos guardarlo, previamente definidas en un man-

tenimiento donde se especifican estos datos.

Primeramente se rellenan los todos datos relacionados con la documentación para el usuario.

Seguidamente se especifican los requerimientos necesarios para la instalación.

Especificamos la documentación necesaria para el implantador/administrador.

Y por último tenemos las funcionalidades corregidas que se genera de manera automática a través de las CRs del producto y la versión indicados en la cabecera y sobre el cual estamos haciendo el Release notes. Las CRs son "hyperlinks" que nos llevan al "Registro de componentes" para poder modificar las funcionalidades corregidas.

Toda esta información nos acaba generando un documento en formato HTML.

8.8 Generación automática de órdenes de archivo.

En este mantenimiento se especifican el producto y versión y todos los kits de distribución resultantes que se han generado.

Toda esta información se rellena a través de una plantilla en formato HTML.

8.9 Generación automática del dis0.

Como en el caso anterior, en este mantenimiento a través del producto y la versión, se especifican los campos necesarios en este apartado para generar el DIS0.

Esta información rellena una plantilla en formato HTML.

8.10 Creación de plantillas con formato HTML de la documentación de una distribución.

Como he comentado anteriormente, las órdenes de archivo y los DIS0 se acaban generando en un documento con formato HTML.

Estos documentos inicialmente estaban en formato Word, pero dada su complejidad y dificultad a la hora de automatizarlos se decidió renovarlos y crearlos siguiendo la estructura de un documento de UNIT4 en formato HTML.

8.11 Automatismos en los cierres de versiones.

Cuando una versión de un producto ya ha sido distribuida se debe cerrar. Para esto se ha automatizado este proceso, donde al cerrar una versión ya no permite hacer más carga de componentes ni a través de repositorio ni de documentos Excel.

Además de permitirte generar la documentación referente a ese producto y versión (reléase notes, dis0 y órdenes de archivo) de manera automática y a la vez sin tener que generarlos uno por uno en los diferentes mantenimientos.

8.12 Documentación generada.

Todos los documentos y plantillas generados y resultantes de los diferentes mantenimientos/formularios realizados en cada uno de los objetivos anteriores del proyecto y las imágenes de la aplicación, se pueden observar en el tanto en el Dossier del TFG, en el documento del Informe de Proyecto II, en la carpeta de Otros documentos y en el Anexo.

9 Beneficios

Entre los beneficios esperados del proyecto se podría destacar principalmente uno, el de unificar en una sola aplicación todas las necesidades y características de cada grupo de trabajo a la hora de registrar componentes. Esto genera otros beneficios como el ahorro en costes y tiempo así como el aumento de la calidad y de la fiabilidad de los kits de distribución.

También se prevé un aumento de la producción al tener como un módulo de la aplicación que utilizan todos los grupos a la hora de trabajar, y aumentaría la uniformidad a la hora de obtener plantillas e impresos de las CR efectuadas y a la hora de visualizarlas a través de la aplicación.

Se estima que gracias a la aplicación realizada, se ahorre aproximadamente un día de trabajo por cada distribución.

10 Conclusión

En términos generales, el proyecto se ha completado satisfactoriamente sin ninguna incidencia y según la planificación estimada, realizando incluso los objetivos para la ampliación además de añadir algunos objetivos adicionales en el desarrollo de la aplicación. Debido a esto se decidió cambiar el nombre del proyecto (Registro de componentes) por el actual, ya que el proyecto abarca más de lo que se había previsto en un principio.

Este proyecto aporta a la empresa de UNIT4 un aumento en cuanto a calidad y mejora los métodos actuales de trabajo, cosa que conlleva una mejora sustancial en cuanto a ahorro de tiempo y consecuente aumento de la producción.

Otra conclusión positiva es que su aplicación es directa y de uso inmediato por los diferentes equipos de R&D, que son los usuarios finales de la aplicación, ya que se ha realizado la integración en la aplicación ekon.SCRUM.

La elección de karat como herramienta de desarrollo ha sido extraordinariamente útil en la consecución del proyecto, en cuanto la facilidad de desarrollo como para su integración con el resto de aplicaciones.

Desde el punto de vista práctico, la aplicación es intuitiva y fácil de utilizar.

Desde el punto de vista personal, realizar este proyecto a través de la empresa UNIT4, ha sido una vivencia positiva en cuanto a la experiencia ganada y de la mejoría personal de la cual he sido observador en cuanto a desarrollo, análisis y diseño.

11 Futuras Extensiones

En cuanto a las posibles extensiones, el proyecto es escalable en la medida que sea necesario si surgieran nuevas necesidades por parte de los diferentes equipos R&D de UNIT4.

Se plantean algunos ejemplos:

- Enlazar la generación de plantillas con la generación automática de kits (Jenkins).
- Envío automático de e-mails al cerrar versión (con órdenes de archivos, dis0, releases notes…).

- Automatismos a la hora de obtener los componentes modificados por CR, utilizando studio de karat (necesidad de apoyo de plataforma).

AGRADECIMIENTOS

Me gustaría agradecer a todas las personas que me han ayudado a llevar el proyecto a buen puerto y sin cuya ayuda no habría sido capaz de alcanzar los objetivos deseados.

Primeramente a mi familia por el apoyo brindado durante todos estos años de carrera.

A Josep Lladós como tutor por parte de la universidad, por su enfoque más académico del proyecto y por aconsejar y revisar toda la documentación del proyecto.

A Ramón Torres como tutor en UNIT4 que me ha guiado y me ha asesorado durante mi estancia en la empresa.

Y por último, a todo el equipo de ERP, en especial a José Miguel Cabello y Germán Álvarez por su inestimable ayuda y asesoramiento durante el desarrollo del proyecto. Sin cuya ayuda no habría conseguido alcanzar los objetivos deseados.

BIBLIOGRAFÍA

[1] Xavier Albaladejo, "SCRUM", http://www.proyectosagiles.org/que-es-scrum, 2014.

[2] UNIT4, "Arquitectura de karat", http://www.argon.es/pdf/karat-studio-folleto-es.pdf, 2014.

[3] Microsoft Dynamics CRM, "Prodware Service Desk", http://www.dynamics-crm.es/microsoft-dynamics-crm-service-desk?s=adwords_Service_Desk&gclid=CLLm_fWQ970CFSQFwwodEokAhg.

[4] KMKey, "Knowledge Management Key", http://www.kmkey.com/kmkey_com/es/productos/, 2014.

[5] Nitin Mittal, "The Burn-Down Chart: An Effective Planning and Tracking Tool", http://www.scrumalliance.org/community/articles/2013/august/burn-down-chart-%E2%80%93-an-effective-planning-and-tracki, 2013.

[6] UNIT4, "API specification for the karat Platform, version 9.1", unpublished document.

[7] Oracle Corporation, "What is Java technology and why do I need it?", https://www.java.com/en/download/faq/whatis_java.xml, 2014.

[8] Eclipse, "Who is Kepler?", http://www.eclipse.org/kepler/, June 26, 2013.

[9] SQL SQuirreL, "Universal SQL Client", http://squirrel-sql.sourceforge.net/#overview, 2014.

APÉNDICE

A1. IMÁGENES DE LA APLICACIÓN

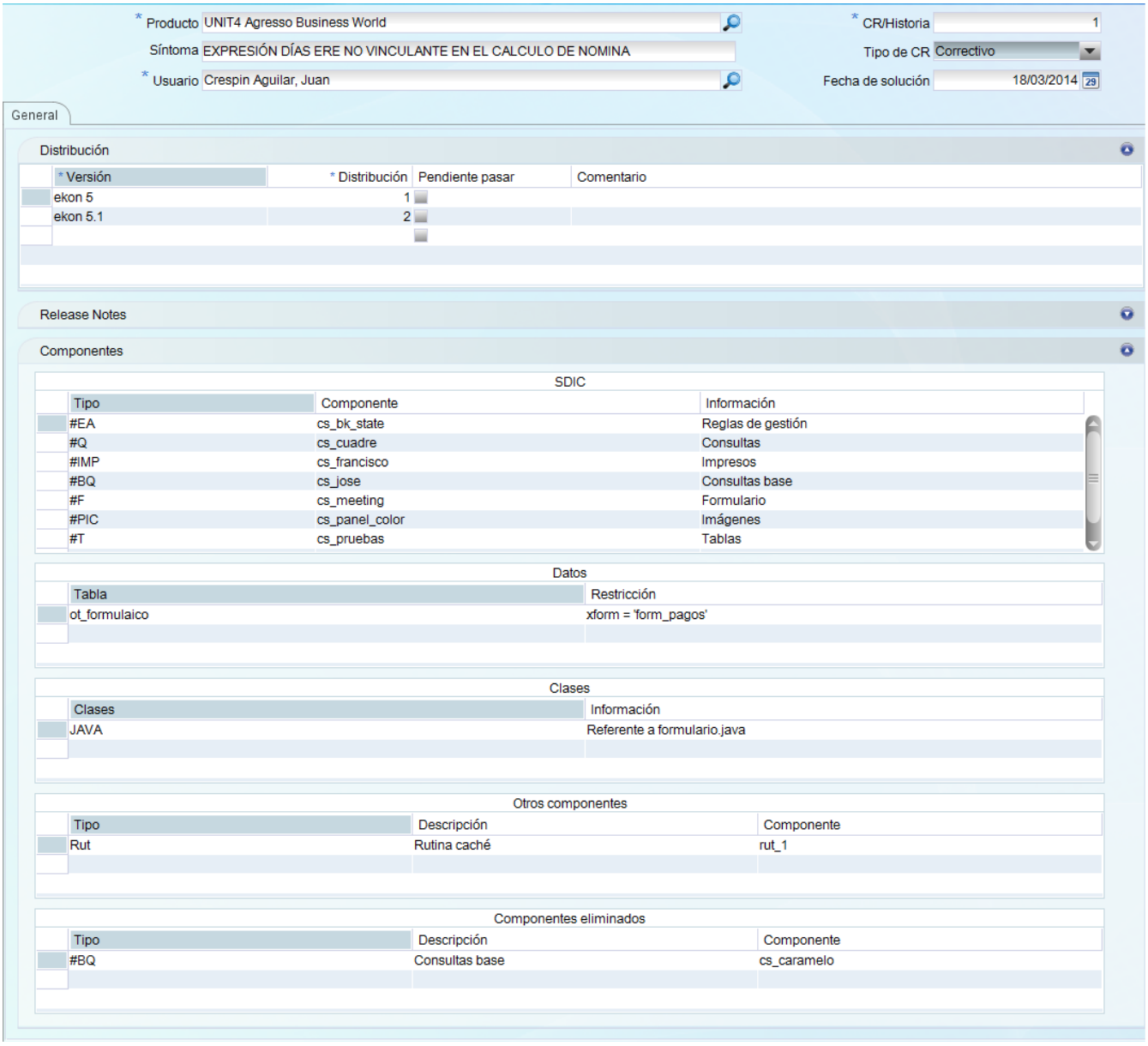

FIGURA 13. REGISTRO DE COMPONENTES POR VERSIÓN.

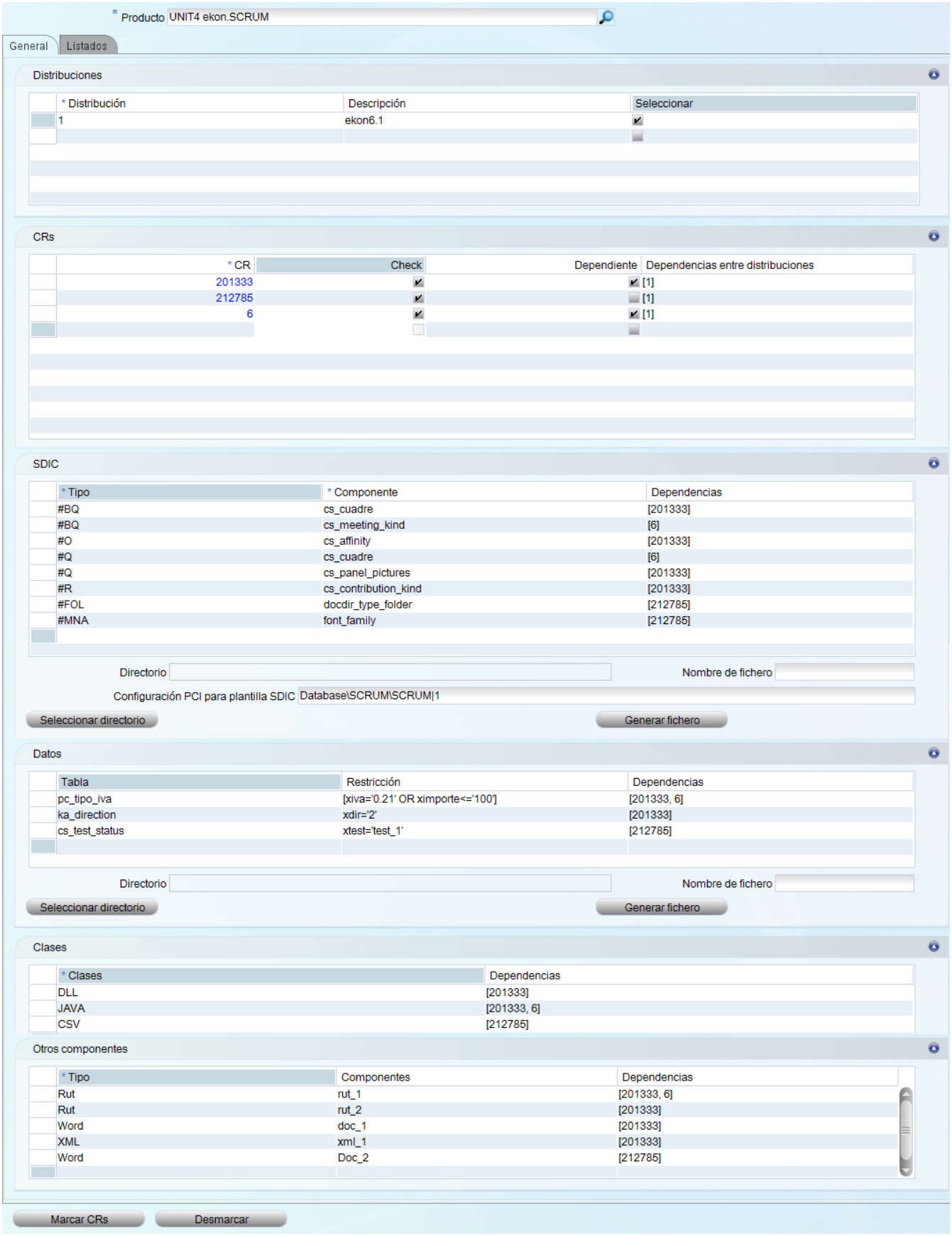

FIGURA 14. BÚSQUEDA DE DEPENDENCIAS Y GENERACIÓN AUTOMÁTICA DE PLANTILLAS PARA LAS DISTRIBUCIONES.

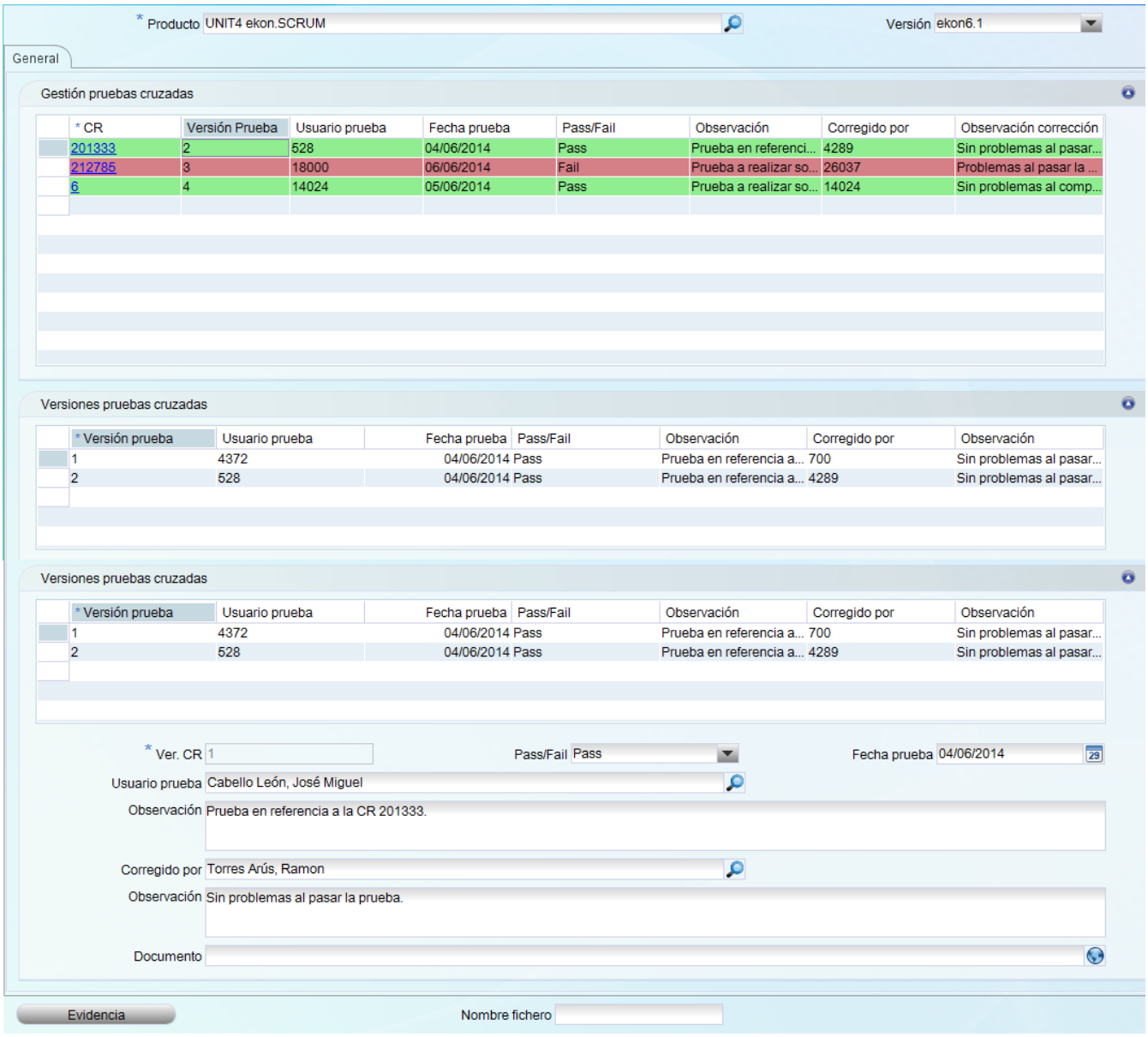

FIGURA 15. PRUEBAS CRUZADAS.

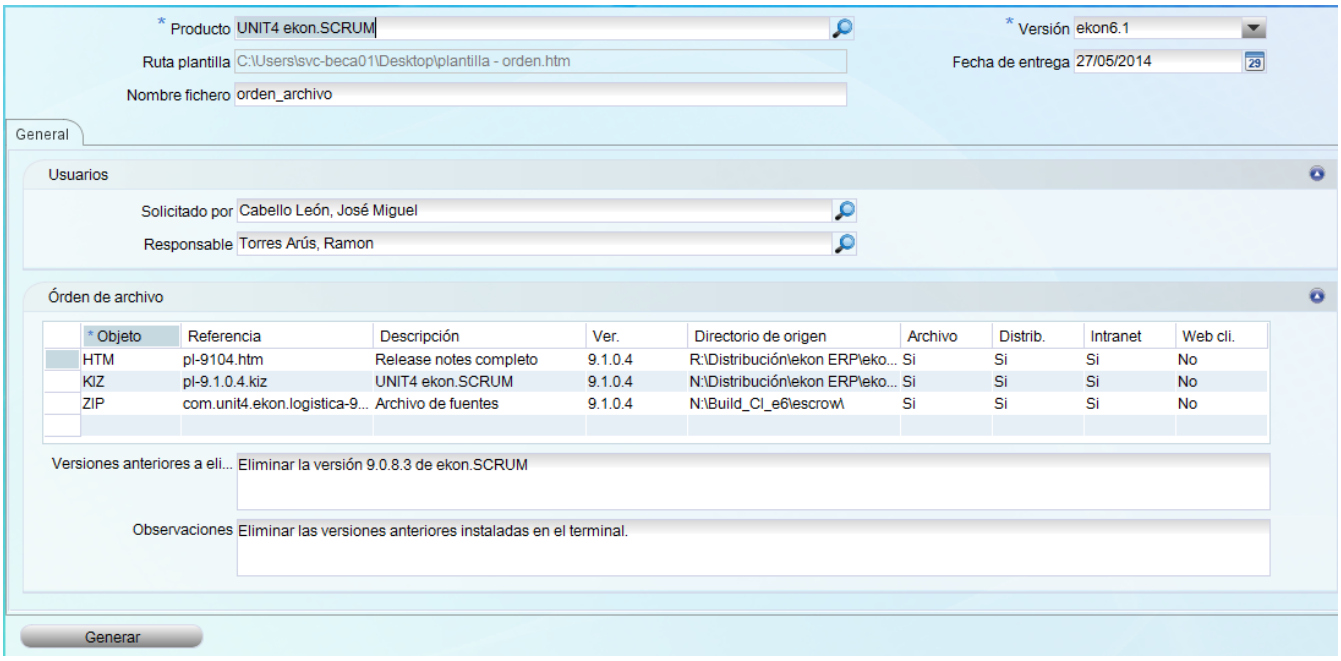

FIGURA 16. ÓRDENES DE ARCHIVO.

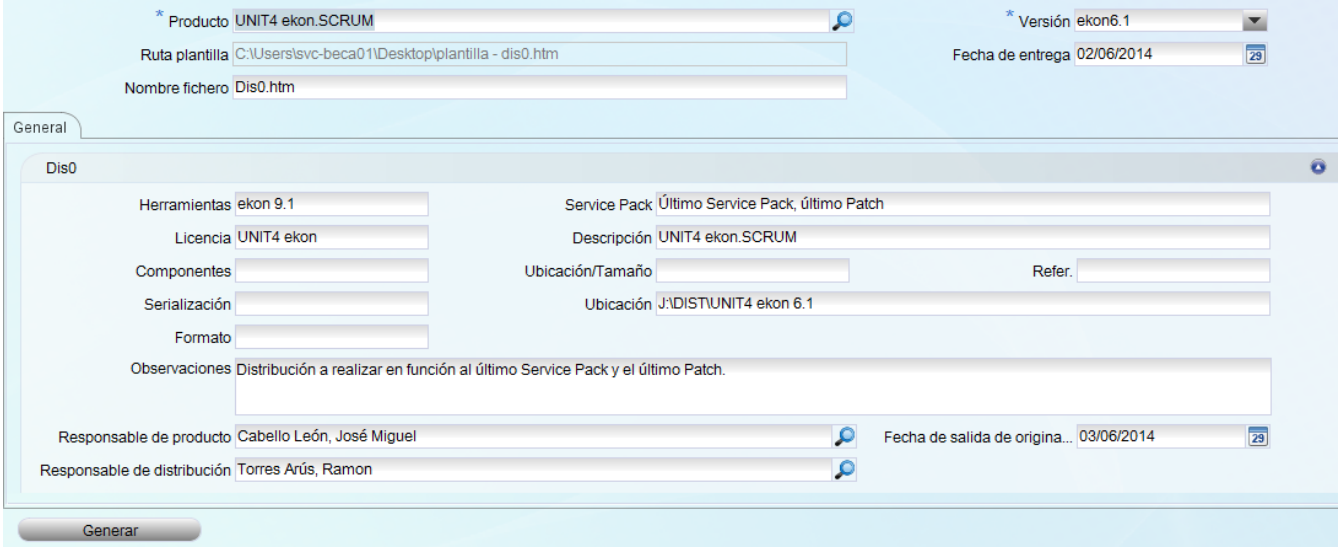

FIGURA 17. DIS0.