UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Estrategias para abordar el mantenimiento de un producto software de gran envergadura

Autor: Carlos Sánchez Rodríguez

Director: Dr. Patricio Letelier Torres

Julio del 2012

Trabajo Final de Máster presentado para cumplir con los requisitos finales para la obtención del titulo de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información, de la Universidad Politécnica de Valencia,2012.

Agradecimientos

Quiero agradecer este trabajo a las personas que me han ayudado tanto físicamente como anímicamente para que pudiera llevar el trabajo a buen puerto.

Un especial agradecimiento a mi director de tesis, Patricio, siempre te recordare como mas que un director, sino como un amigo. He aprendido de ti que un mismo objetivo puede desglosarse en objetivos más pequeños y que las opiniones de los demás importan tanto como la de uno mismo, aun siendo tu el responsable de tomar la decisión.

Agradecer a mis compañeros de laboratorio por ayudarme siempre en la medida de lo posible, por aguantarme y por escucharme y respetar mis opiniones aunque a veces no tuviera razón o fueran contrarias a las de ellos

Agradecer también a la empresa ADD Informática, sin su flexibilidad este trabajo no habría sido posible.

Y como no agradecer a mis padres y a ti. Ellos siempre me han apoyado y han sido ellos los que siempre me han empujado a rendir al 100% en mis actividades, aunque a veces se han llevado la peor parte que un trabajo de este tipo conlleva.

Gracias a todos.

"Rezar para que la buena suerte no se le ocurra abandonarme, no olvidar que me lo he pasado de muerte y que nadie puede frenarme"

Tabla de contenidos

Tabla de Ilustraciones

Capitulo 1. Introducción

1.1. Motivación

"Sin importar en qué momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida."

Primera Ley de Ingeniería de sistemas [16]

El software es cambiante, la tecnología va cambiando y necesidades que no se tenían al inicio del desarrollo de un producto se te presentan cuando el producto está casi terminado.

Cuando se aborda el desarrollo de un nuevo producto se debe realizar con la premisa de que el producto va a sufrir cambios, a veces estos cambios se producen antes de que se vea a luz una primera versión del producto.

Los requerimientos van cambiando y por lo tanto el software debe evolucionar constantemente, a veces los cambios que sufre un producto pueden modificar funcionalidad existente y/o añadir funcionalidad en distintas partes del producto

En estas situaciones donde el producto necesita un cambio hay que tener especial cuidado que durante el cambio no afecte el comportamiento del producto que se quieren conservar.

En TUNE-UP Software Process herramienta que utilizaremos tanto para abordar las nuevas necesidades, como el uso de su metodología en el desarrollo de las mismas, requiere de nuevas necesidades, en concreto en esta tesis vamos a tratar de incluir dos nuevas funcionalidades, la primera de ellas el mantenimiento de pruebas nos servirá para redefinir pruebas dentro del gestor de requisitos y con esto mejoraremos la descripción de la especificación y/o añadiremos pruebas que no estuvieran definidas previamente. La segunda funcionalidad nos permitirá la gestión de proyectos, esto nos facilitara las tareas de coordinación de equipos de trabajo de un proyecto así como la integración de las unidades de trabajo(WU) con los proyectos.

1.2. Objetivos

En el desarrollo de la tesis se van a desarrollar dos desafíos que aportaran técnicas y funcionalidad a la metodología y a la herramienta TUNE-UP Software Procces.

El primer desafío consiste en el mantenimiento de las pruebas de aceptación incluidas dentro del gestor de requisitos de la herramienta. Esto permitirá a los analistas refactorizar[1] pruebas que no estaban correctamente definidas y/o añadir comportamiento del sistema que existe pero que no está definido por pruebas en el gestor de requisitos.

El segundo desafío consiste en el desarrollo de un gestor de proyectos integrado dentro de la herramienta. Esto constituye una nueva funcionalidad para la coordinación de varios equipos trabajando en el mismo proyecto. Este desafío establecerá un cambio del concepto de proyecto dentro de la herramienta además de la inclusión de una técnica que facilita esa coordinación en los inicios del proyecto.

1.3. Estructura del Trabajo

La tesis está dividida en 6 capítulos. Dentro de los 6 capítulos se pueden diferenciar dos bloques, en cada uno de los bloques se desarrolla un nuevo desafío, a continuación se describe el contenido de cada uno de los capítulos.

El capítulo 1, se presenta la motivación a la que nos ha llevado el desarrollo de esta tesina así como los objetivos de la misma.

El capitulo 2 explica los aspectos básicos de la metodología TUNE-UP y de su herramienta de apoyo TUNE-UP Software Procces. Es sobre esta herramienta donde se desarrollaran los desafíos por lo que es importante conocer como se trabaja dentro de la herramienta.

[1] El termino refactorizar se utiliza en el ámbito del código para realizar mejoras en la estructura del código. En esta tesis extrapolamos el concepto al ámbito de los requisitos expresados como PAs, con lo que la palabra nos sirve para definir la acción de mejorar la especificación de requisitos sin cambiar el comportamiento

En el capítulo 3 forma parte del primer desafío, el mantenimiento de las pruebas de aceptación. El capitulo presenta se presenta el enfoque Test Driven Requirements Engineering de TUNE-UP. Esta metodología es el marco de trabajo para desarrollar una solución para los desafíos propuestos en esta tesis. Además se detalla de forma precisa el proceso dirigido por las Pruebas de Aceptación y cómo trabajar de esa forma aporta beneficios a la hora de hacer el seguimiento de una iteración. Al final del capítulo se describe el proceso de desarrollo para cumplir el objetivo del primer desafío, un sistema que permita el mantenimiento de pruebas actuales del sistema.

El capitulo 5 forma parte del segundo desafío, la gestión de proyectos dentro de la herramienta TUNE-UP. El capitulo se divide en tres partes: la primera parte se introducen los proyectos y el concepto de proyecto desde TUNE-UP. La segunda parte se centra en el desarrollo de la nueva gestión de proyectos dentro de la herramienta TUNE-UP, esta parte además incluye una técnica de coordinación de proyectos y la tercera parte se plantean dos técnicas de visualización de proyectos.

El capitulo 7 se concluye el trabajo mostrando las conclusiones obtenidas durante el desarrollo del mismo, también se plantean algunas posibilidades de trabajo futuro.

Capitulo 2. TUNE-UP

El resumen de TUNE-UP presentado en este capítulo se ha elaborado a partir del contenido de la Tesis de María Company Bria)[1]

2.1. Metodología TUNE-UP

TUNE-UP es una metodología desarrollada a partir del trabajo día a día en una PYME de desarrollo de software con vocación de mejora continua del proceso. TUNE-UP se ha definido bajo el marco de varios proyectos universidad-empresa. TUNE-UP ha conseguido su madurez en más de seis años de aplicación y refinamiento aplicándose en proyectos industriales, en el ámbito académico y también como objeto de publicaciones de investigación.

TUNE-UP se inspira en la esencia de **PSP** (Personal Software Process) [12], donde se destaca que la base del éxito radica en una disciplina de trabajo y productividad individual centrada en la gestión de los compromisos. En cuanto a gestión del tiempo y de priorización del trabajo personal TUNE-UP está alineada con los principios de **GTD** (Getting Things Done)[13] y **The Seven Habits of Highly Effective People** [14]. Para la organización y fácil acceso al trabajo de un agente TUNE-UP aporta una innovadora variación de **sistema Kanban**, el cual se integra con un simple pero potente mecanismo de workflows flexibles, permitiendo orquestar de forma automática gran parte de la colaboración necesaria entre actividades. TUNE-UP ayuda en cada momento del proyecto a responder a la pregunta: *¿conseguiré cumplir con los plazos de entrega de mis tareas? o ¿seremos capaces de cumplir con los plazos de entrega al cliente?* Estas simples preguntas inquietan a cualquier gestor o participante de un proyecto. No contar con una respuesta acertada, y sobre todo oportuna, conlleva en la mayoría de los casos graves complicaciones en el proyecto.

¿Por qué otra metodología y otra herramienta existiendo ya alternativas disponibles?

Después de trabajar tanto con metodologías tradicionales **(RUP y Métrica)** como ágiles **(XP y SCRUM)**, y utilizado muchas herramientas de apoyo para diferentes actividades en el proceso de desarrollo (herramientas para modelado o para pruebas, IDEs para

programación, etc.), la principal lección aprendida es que los lenguajes de programación y notaciones, así como sus herramientas de apoyo, aunque contribuyen al éxito de un proyecto no son tan decisivos como lo es el proceso de desarrollo en sí. Sin embargo, siendo las metodologías las protagonistas al respecto, las propuestas actuales continúan lejanas de la realidad cotidiana de un equipo de desarrollo. Por un lado, las metodologías tradicionales de entrada tienen el impedimento de ser demasiado genéricas y amplias, lo cual requiere un gran esfuerzo de concepción e implantación en el equipo, el cual puede ser inviable de abordar. Por otra parte, las metodologías ágiles parten de una visión demasiado simplificada de lo que es un proceso de desarrollo y particularmente del mantenimiento. Un producto exitoso requerirá mantenimiento. Con una metodología ágil el proceso de generación inicial de un producto puede resultar acelerado pero planteándose una perspectiva seria de mantenimiento no constituye un enfoque adecuado, puesto que cada vez se va haciendo más imprescindible el llevar un registro y control de los cambios y del comportamiento implementado en el producto. Esta información en las metodologías ágiles no es gestionada (ni siquiera almacenada) después de la implementación. Normalmente las metodologías tradicionales proponen una planificación también tradicional, es decir, basada en las técnicas genéricas utilizadas en todo tipo de proyectos. Dichas técnicas están siendo duramente criticadas y quedan en evidencia sus carencias para realizar un seguimiento del proyecto cuando el modelo de proceso es iterativo e incremental. *¿Quién ha probado con éxito, en este contexto, gestionar adecuadamente un proyecto software usando un diagrama Gantt?* Las metodologías ágiles han sabido abordar este aspecto y uno de sus puntos más destacables es precisamente la planificación, utilizando técnicas tales como los **diagramas Burn-down**.

Un proyecto de desarrollo o mantenimiento de software tiene por objetivo el conseguir una entrega exitosa del producto, es decir, una versión operacional que pueda ser explotada y rentabilizada por el cliente. Para conseguir este objetivo TUNE-UP incluye los siguientes elementos:

- **Modelo de proceso iterativo e incremental** para el desarrollo y mantenimiento del software. El trabajo se divide en Unidades de Trabajo(WUs) que son

asignadas a versiones. Las versiones son frecuentes y de corta duración, entre 3 y 6 semanas dependiendo del producto.

- **Proceso de desarrollo dirigido por las pruebas** (Test-Driven). La definición de una WU es básicamente la especificación de sus pruebas de aceptación acordadas con el cliente. A partir de ahí, todo el proceso gira en torno a ellas, se estima el esfuerzo de implementar el comportamiento asociado y de aplicar las pruebas, se diseñan las pruebas e implementa dicho comportamiento, y finalmente, se aplican las pruebas sobre el producto para garantizar su correcta implementación.

- **Workflows flexibles para la coordinación del trabajo asociado a cada unidad de trabajo**. Los productos, según sus características, tienen asociados un conjunto de workflows que son utilizados para realizar WUs. Cada WU sigue el flujo de actividades del workflow. Bajo ciertas condiciones se permite saltar hacia adelante o hacia atrás en el workflow, así como cambios de agentes asignados e incluso cambio de workflow. Por ejemplo, las típicas situaciones de re-trabajo en desarrollo de software ocasionadas por detección de defectos pueden originar saltos atrás no explícitos en el workflow. Otras facilidades en cuanto a flexibilidad es permitir trabajo en paralelo o incluso añadir actividades no contempladas en la definición del workflow. Esta flexibilidad a su vez evita que la especificación del workflow se complique añadiendo explícitamente todas las situaciones posibles. Los workflows en TUNE-UP actúan como guías esenciales del proceso pero evitando las rigideces usuales de la tecnología y herramientas específicas en el ámbito de workflows. En los workflows y mediante su refinamiento se va plasmando la mejora continua de proceso.

- **Planificación y seguimiento continuo**. En todo momento debe estar actualizado el estado de las versiones, de las WU, y del trabajo asignado a los agentes. El jefe del proyecto puede actuar oportunamente con dicha información, tomando decisiones tales como: negociar el alcance de la versión con el cliente, conseguir más recursos, redistribuir carga de trabajo entre agentes, cambiar los plazos de la versión, mover WUs entre versiones, etc.

- **Control de tiempos**. Los agentes registran el tiempo que dedican a la realización de las actividades, el cual se compara con los tiempos estimados en cada una de ellas, detectando oportunamente desviaciones significativas. Esto permite a los agentes gestionar más efectivamente su tiempo, mejorar sus estimaciones y ofrecer al jefe del proyecto información actualizada del estado de la versión.

TUNE-UP es una metodología que incorpora aspectos de metodologías ágiles y de metodologías tradicionales. Las dos primeras características (proceso iterativo e incremental, y proceso centrado en las pruebas de aceptación) clasifican a TUNE-UP como metodología ágil, sin embargo, las otras características están más próximas de lo que sería una metodología tradicional.

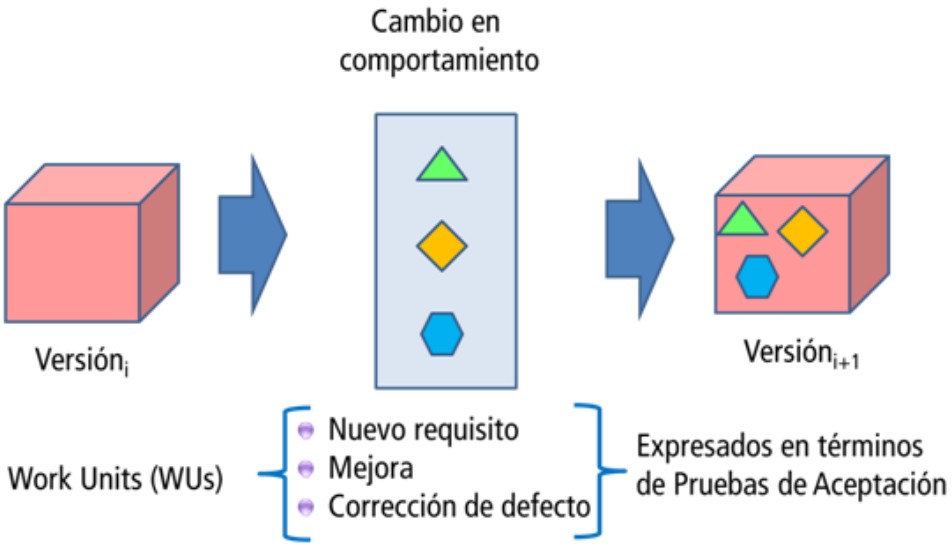


Ilustración 1. Estrategia Global de TUNE-UP

La Ilustración 1ilustra la estrategia global de desarrollo propuesta por TUNE-UP. Una nueva versión del producto incluirá cambios de comportamiento con respecto de la versión anterior. Dichos cambios de comportamiento pueden ser en general **Nuevos Requisitos**, **Mejoras de requisitos** existentes o **Correcciones de defectos**. En TUNE-UP todos estos tipos de cambios de comportamiento se especifican de forma homogénea como PAs. Así, por un lado desde el punto de organización del trabajo todo tipo de cambio que está en una versión se aborda de forma integrada en cuanto a planificación y asignación de recursos. Por otra parte, desde la perspectiva de la

especificación del cambio y su posterior verificación, en lugar de tener artefactos diferentes (requisitos/cambios/correcciones y PAs), se tiene sólo un artefacto llamado **Unidad de Trabajo** (WorkUnit, WU) expresado en términos de PAs.

TUNE-UP está dirigido por las PAs. Es un enfoque TDD pero en el ámbito más general de la gestión del proyecto y especialmente de los requisitos del producto. Tal como se ilustra en la Ilustración 2, y a modo de ejemplo (pues los roles podrían variar según sean los recursos disponibles y las exigencias del proyecto), el Analista define con el Cliente los cambios en el producto en términos de WUs y sus correspondientes PAs. El Programador escribirá código para satisfacer dichas PAs y finalmente el Tester aplicará pruebas para asegurar que el comportamiento del sistema satisface las PAs.

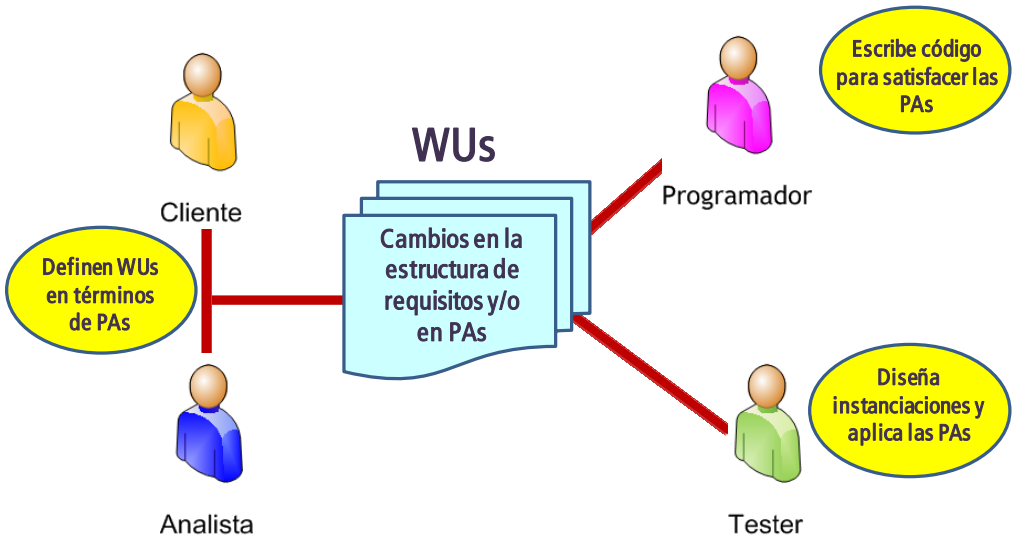


Ilustración 2. Proceso dirigido por las PAs

Dependiendo del tipo de WU y del proyecto en el cual debe llevarse a cabo el cambio de comportamiento de un producto, puede ser necesario realizar diferentes actividades y en diferente ordenamiento temporal. En TUNE-UP cuando se crea una WU se le asigna el workflow más apropiado. La Ilustración 3muestra un **workflow** básico para el desarrollo de una WU, en el cual se han definido cuatro roles y diez actividades. TUNE-UP permite crear y modificar workflows según se requiera. En TUNE-UP se pueden gestionar diferentes productos con diversos equipos de desarrolladores. Cada producto tiene asociado ciertos workflows los cuales pueden compartirse entre diferentes productos, así como las actividades puedes ser utilizadas en diferentes workflows. Las actividades de cada workflow pueden variar

significativamente dependiendo de factores tales como: cantidad y especialización de agentes participantes, validaciones o negociaciones predeterminadas con el cliente, características del producto (necesidad de migración, traducción, etc.), niveles y actividades de pruebas (unitarias, de integración, de aceptación, pruebas de regresión, automatización de pruebas), etc. El proceso de mejora continua se plasma en el refinamiento de los workflows.

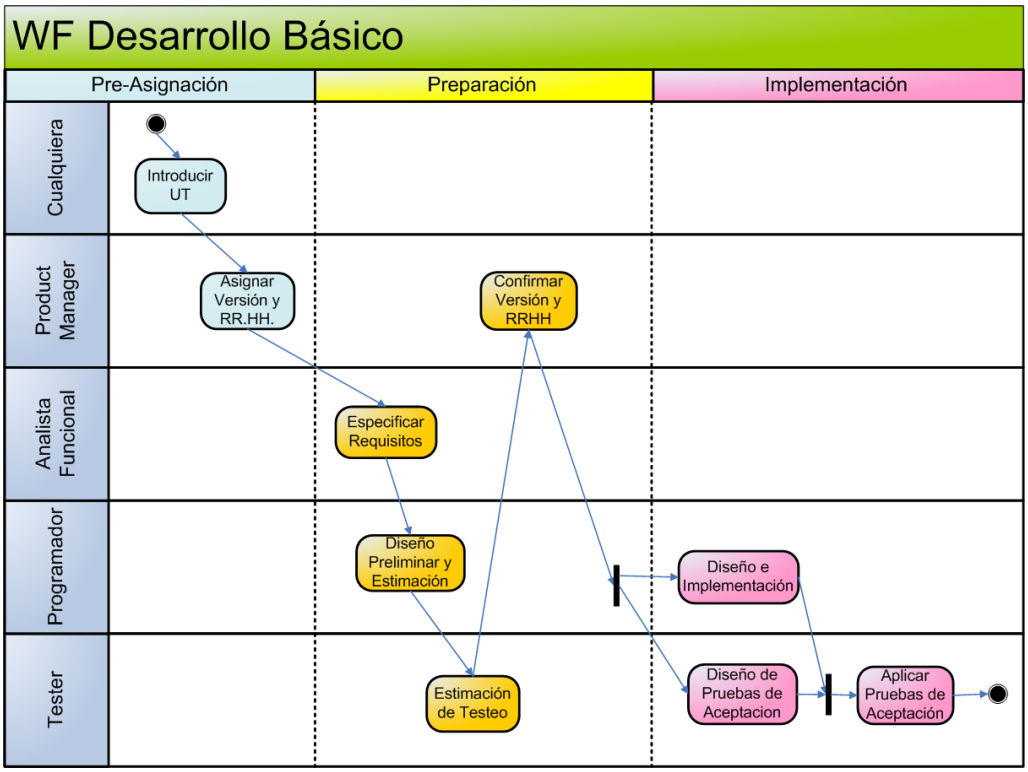


Ilustración 3. Un ejemplo de Workflow simple

En TUNE-UP se utiliza la siguiente terminología para referirse a comportamiento no deseado del producto: Fallo, Defecto y Error. El **fallo** es el síntoma de comportamiento no deseado que manifiesta el producto en su utilización. El **defecto** es lo que produce el fallo. Los defectos pueden estar en el código del producto o en cualquiera de sus especificaciones, por ejemplo puede que el código esté correcto respecto de la prueba, pero la prueba puede no ser correcta. Finalmente, el **error** es la acción humana (realizada o no realizada por los desarrolladores), es lo que originó el defecto. En la Ilustración 4 se ilustra cómo en TUNE-UP se lleva a cabo un tratamiento integral de los fallos. Todo fallo se especifica como una PA con un KO contenida en una WU. En caso que el fallo se resuelva en una misma WU (sea esta de cualquier tipo), se llevará a cabo

el correspondiente re-trabajo (volviendo a realizar el análisis o programación requerida). Si el fallo no es abordado en la WU donde se detecta simplemente si la detección del fallo se hace fuera del contexto de una WU, entonces se creará una nueva WU de tipo fallo que se llevará a cabo en la versión actual o en alguna versión futura.

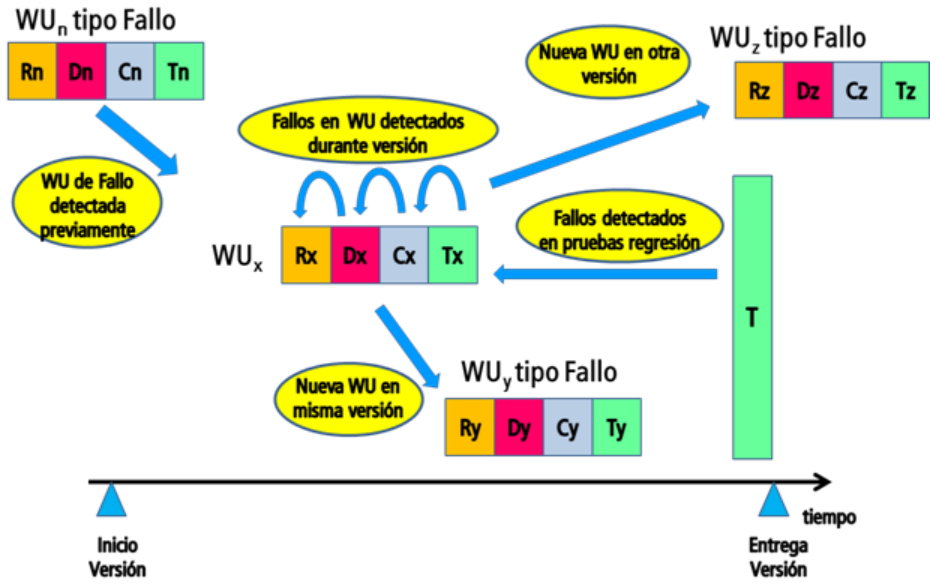


Ilustración 4. Gestión de Fallos en TUNE-UP

2.2. Herramienta TUNE-UP Software Process

TUNE-UP Software Process es el nombre de la herramienta de apoyo para la aplicación efectiva de la metodología TUNE-UP. La herramienta está formada por cuatro módulos principales: **Personal Planner (PEP)**, **Work Unit Manager (WUM)**, **Version Contents & Tracking (VCT)** y **Requirements Manager (REM)**. A continuación se describe brevemente cada uno de estos módulos..

Personal Planner (PEP)

Es el punto de entrada al espacio de trabajo del agente. Cuando un agente inicia su jornada laboral, accede al PEP (Ilustración 5) para ver el trabajo en el cual participa. Este trabajo corresponde a las actividades que tenga asignadas en las WUs no terminadas. En el **Kanban** (grid del extremo superior izquierdo de la Ilustración 5) el

agente puede determinar fácilmente en el PEP qué actividades tiene pendientes o en progreso (activas o pausadas). Además puede distinguir aquellas actividades que tienen pendientes o en proceso otros miembros del equipo y que, o bien estarían por llegarle, o ya ha finalizado, y han continuado en actividades posteriores en el workflow. El agente tiene que seleccionar la WU en la qué va a trabajar. Un aspecto clave para el proyecto es que los agentes se dediquen a las actividades acertadas de acuerdo a sus prioridades. El PEP ofrece una variedad de facilidades de filtrado y ordenamiento de información para que el agente pueda determinar las prioridades de su trabajo y realizar una correcta elección. El Kanban resume las contabilizaciones de las WUs según la actividad y estado en el que se encuentran, y en el grid de la derecha de la Ilustración 5, muestra información de las actividades de dichas WUs incluyendo: producto, versión, descripción de la unidad de trabajo, estado de la actividad actual dentro del workflow, etc.

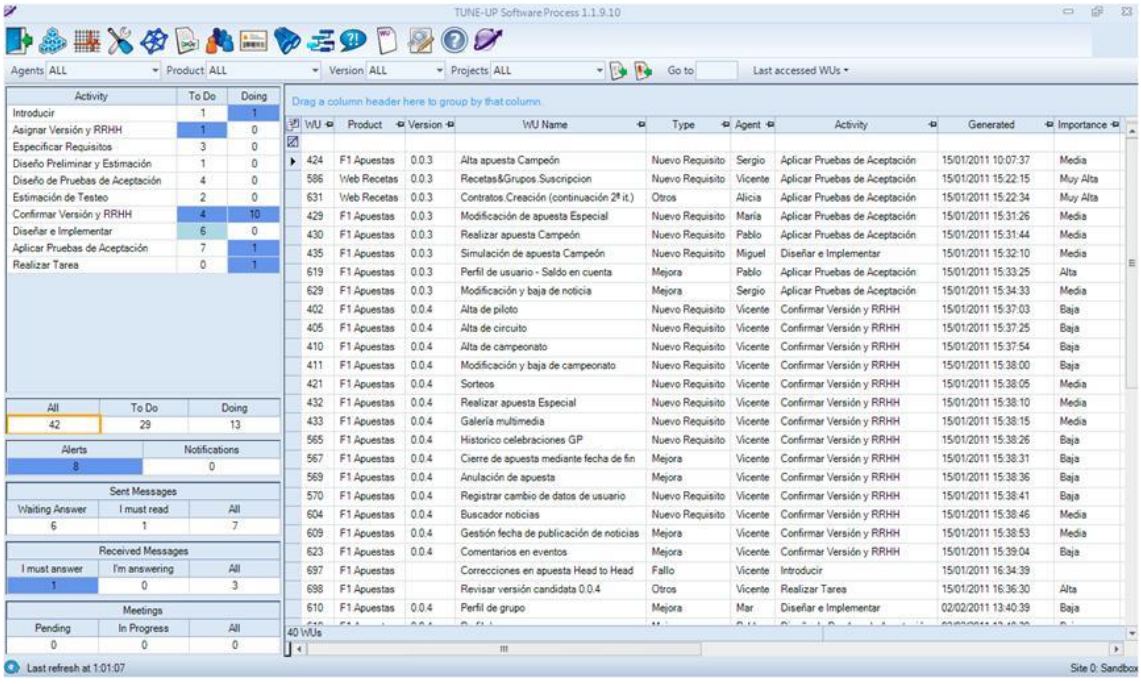


Ilustración 5. Personal Planner (PEP)

Work Unit Manager (WUM)

Cuando el agente decide la WU y la actividad en la que desea trabajar, accede con ella al **WUM**(Ilustración 6). En la ficha de la parte superior del WUM se muestran los datos generales de la WU, y en la parte inferior, un conjunto de pestañas con información más específica. En la **pestaña Tracking** se muestra la lista de actividades del workflow

por las que ha pasado la unidad de trabajo. Cada actividad está asociada a un registro de seguimiento que incluye: la actividad, el agente que la desarrolla, el estado en el que se encuentra, etc. Además, en esta pestaña, con los **botones Record, Pause y Finish**, el agente puede controlar el registro de tiempos, activando, pausando o finalizando la actividad. Al finalizar una actividad, la WU continúa su workflow o puede a petición del usuario dar un salto a otra actividad del worflow. Otras facilidades para gestionar una WU y que se encuentran en otras pestañas del WUM son: **documentación** (para la WU, para apartados específicos del producto, para mensajes como adjuntos y para ejecuciones de PAs por ejemplo para indicar pasos de reproducción de un fallo), **mensajes** para comunicarse con otros agentes en el contexto de la WU, **reuniones** para registrar tiempo y acuerdos alcanzados, **relaciones** con otras WUs, detalle de los **tiempos registrados** y **estimados** para cada actividad, e información de las **partes del producto que son afectadas por la WU**.

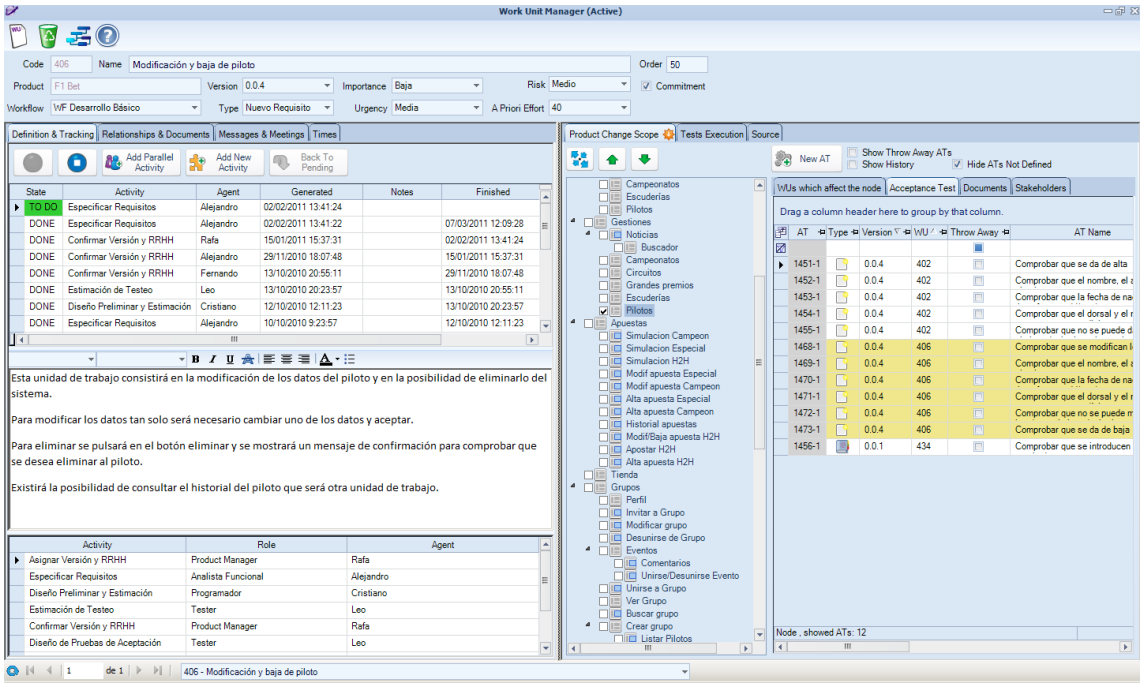


Ilustración 6. Work Unit Manager (WUM)

Requirements Manager (REM)

El **REM** es una de las innovaciones que ofrece TUNE-UP en cuanto a gestión del producto y sus requisitos. En TUNE-UP, el desarrollo y mantenimiento del producto está dirigido por las PAs que constituyen la especificación del comportamiento del producto. La estructura del producto es un **grafo acíclico dirigido** mostrado como un

TreeView, como se muestra en la parte izquierda de la Ilustración 7. Cada nodo de esta estructura es un contenedor de comportamiento expresado como PAs. Así, al seleccionar un nodo, en la **pestaña 'WUs which affect the node'**, podemos conocer toda su historia en términos de WUs que lo han afectado, lo están afectando o lo afectarán en futuras versiones. De manera similar, en la **pestaña 'Acceptance Test'** se puede consultar las PAs que han sido definidas en dicho nodo y cómo han cambiado. Además, cada nodo en la **pestaña 'Documentación'** puede tener asociada documentación adicional en términos de ficheros de cualquier tipo. Finalmente cada nodo en la **pestaña 'Stakeholders'** puede tener especificados los stakeholders para los cuales dicho nodo es relevante. A lo largo de este trabajo se explicara en más detalle este módulo ya que es esencial en la propuesta que se presenta en este trabajo.

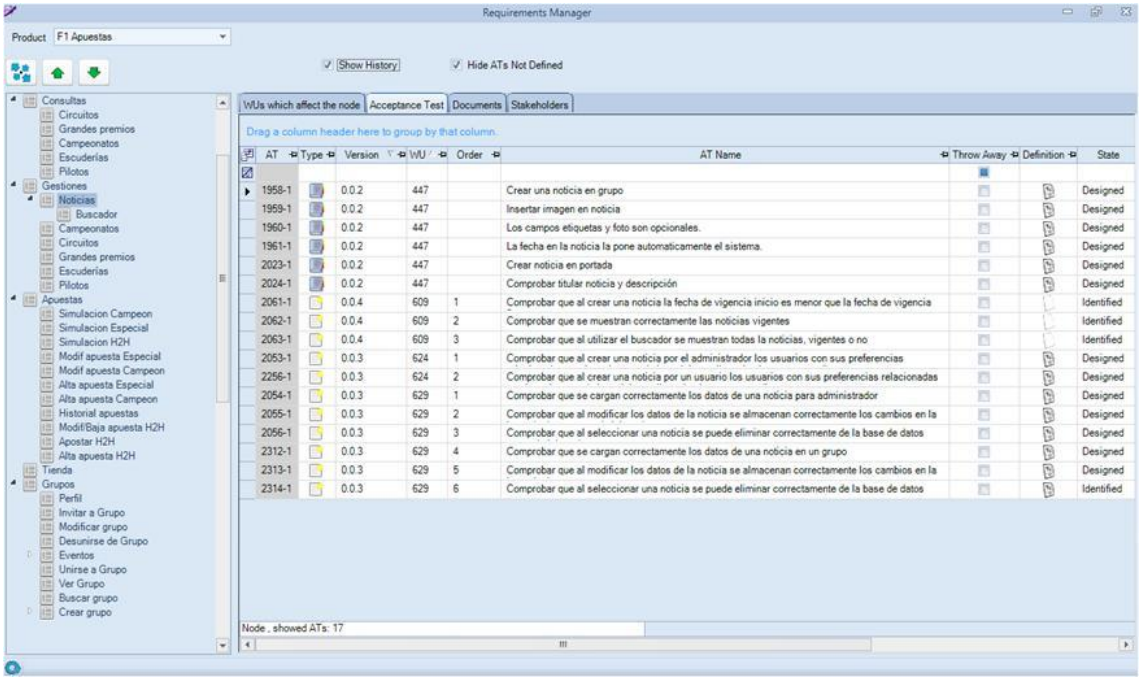


Ilustración 7. Requirements Manager (REM)

Version Contents & Tracking (VCT)

En este módulo encontramos facilidades para la planificación y seguimiento de las versiones. En la Ilustración 8 se muestra el contenido de **la pestaña 'Agent Workload'**, en ella se puede conocer en cualquier momento la holgura simple de los agentes respecto de sus actividades en una versión del producto. En esta interfaz se ofrecen potentes mecanismos de filtros y agrupaciones por columnas. Cuando una versión tiene problemas de holgura, en esta misma interfaz el **Product Manager** puede

cambiar el agente asignado a la actividad (para balancear la carga de un agente) o cambiar de versión alguna unidad de trabajo. Otras alternativas son modificar la fecha de término de la versión, asignar más recursos humanos a la versión o dividir WUs para realizarlas incrementalmente en varias versiones. Además disponemos de otras pestañas que tienen gran utilidad a la hora de planificar una versión como **'Relationships'**, **'WUs in Version'** y **'Affected Requirements'**.

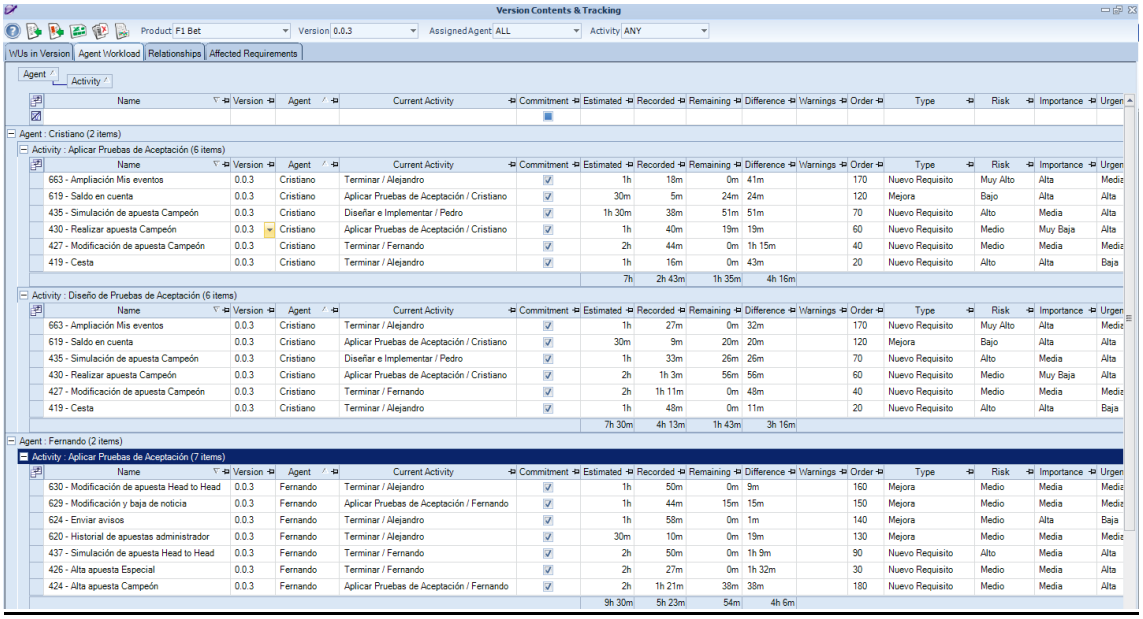


Ilustración 8. Versión Contents & Tracking (VCT)

2.3. Seguimiento de una iteración con TUNE-UP

En TUNE-UP el seguimiento de una iteración incluye los siguientes mecanismos de apoyo:

- **Panel Kanban** en el cual se sintetizan todas las actividades de los workflows en las cuales cada miembro del equipo tiene trabajo asignado. En el Kanban se puede visualizar en qué actividades se encuentran las WUs de una iteración.
- **Módulo *Version Contents and Tracking* (VCT)**. En este módulo se ofrecen varias vistas del contenido y estado de las WUs en una iteración.

- **Alertas y notificaciones a los miembros del equipo.** Automáticamente se generan alertas y notificaciones ante ciertos eventos, por ejemplo, cuando el esfuerzo invertido sobrepasa el esfuerzo estimado en una actividad de una WU, cuando se sobrepasa el tiempo de postergación definido para una actividad, cuando se cambia de versión una WU, etc.

- **La Gráfica Burn Down** es un mecanismo protagonista en el seguimiento de la iteración en TUNE-UP, el cual explicaremos en detalle a continuación.
- **La Gráfica WUs Finished/Unfinished** (lateral inferior izquierdo de la Ilustración 9) muestra por día las WUs que tenemos finalizadas y las que no están finalizadas que cumplen los filtros que se encuentran en la parte superior de laIlustración 9.
- **La gráfica de estado de PAs** (lateral inferior derecho de la Ilustración 9) ilustra el estado de las PAs para cada nivel de testeo de las WUs que cumplen los filtros que se encuentran en la parte superior de la Ilustración 9.

TUNE-UP ofrece una **Gráfica Burn Down** para cada iteración de un producto, junto con una tabla llamada **Daily Events** con información complementaria para la correcta interpretación de la gráfica (esta tabla se explica en detalle más adelante).

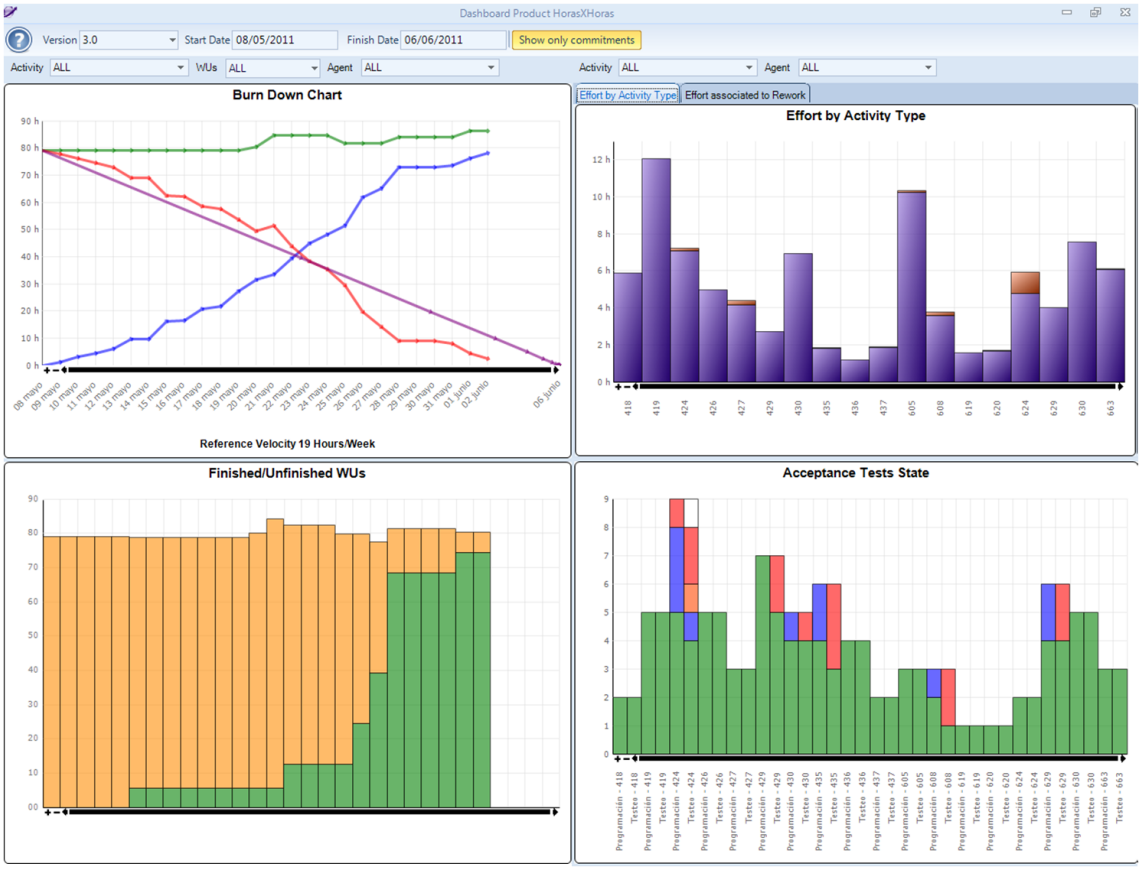


Ilustración 9. Dashboard asociado a un producto

La **Gráfica Burn Down de TUNE-UP** (lateral superior izquierdo de la Ilustración 9) incluye dos gráficas en una; una gráfica básica **Burn Down** (asociada a la línea serpenteante descendente) y una gráfica **Burn Up** (asociada a la línea ascendente representando el esfuerzo invertido). Además, se incluye una línea que representa el **esfuerzo estimado** y una línea diagonal que representa el **esfuerzo restante** de referencia. Gracias a disponer en una misma gráfica de los esfuerzos estimados, invertidos y restantes, se facilita la interpretación del estado de la iteración y cómo se ha ido desarrollando. Situaciones tales como una bajada o subida pronunciada del esfuerzo restante pueden visualmente explicarse por una correspondiente bajada o subida en la línea de esfuerzo estimado, o bien en una subida o bajada en la línea de esfuerzo invertido. Sin embargo, la confirmación de estas interpretaciones, como veremos a continuación, exige contar con la información detallada de los eventos que pueden haber ocurrido entre dos puntos consecutivos de la gráfica. El esfuerzo restante de referencia corresponde a la línea que se traza desde el punto de mayor esfuerzo restante hacia el punto de esfuerzo restante 0 en el día de fin de la iteración.

Además, asociada a esta línea se muestra en la parte inferior de la gráfica la velocidad requerida para conseguir la tendencia ilustrada por ese esfuerzo restante de referencia. Toda la información de la Gráfica Burn Down puede ser filtrada por Actividad, WU y/o Miembro del equipo.

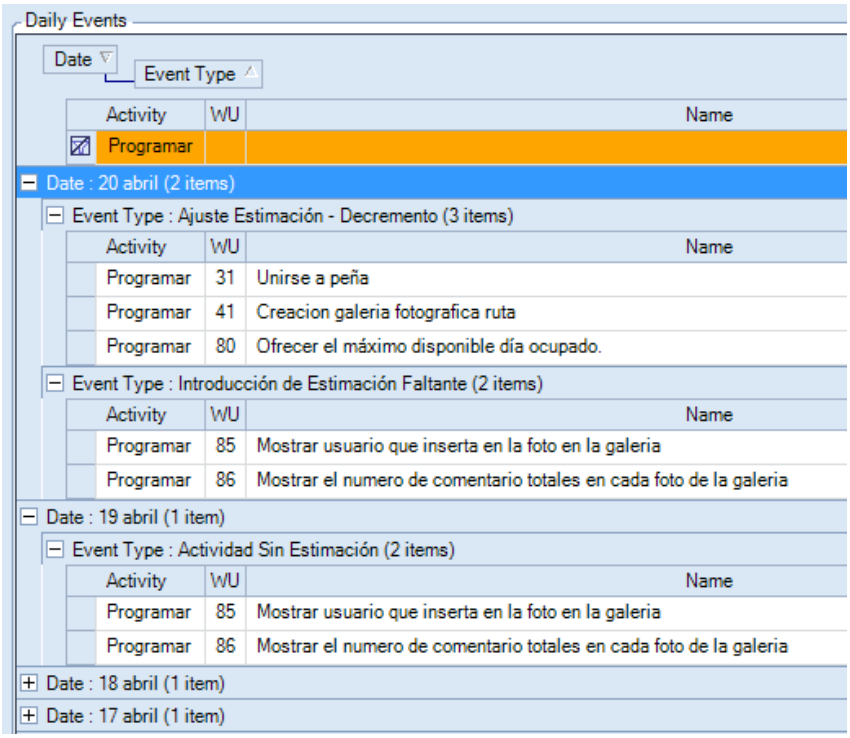


Ilustración 10. Tabla Daily Events

La **Tabla Daily Events** que muestra la Ilustración 10, contiene los eventos diarios que permiten interpretar correctamente la Gráfica Burn Down. Haciendo clic en un punto de la gráfica de la Ilustración 9 se despliega en la tabla de la Ilustración 10 la lista de eventos ocurridos entre el día previo y el día seleccionado. Para cada evento se indica el miembro del equipo, la actividad e información de la WU donde se produce. En TUNE-UP se supervisan todos los eventos que pueden influir en la correcta interpretación del esfuerzo restante, dichos eventos se describen a continuación:

Eventos que invalidan la lectura del esfuerzo restante:

- **Actividad con estimación sobrepasada**: el esfuerzo invertido por el miembro del equipo en la actividad sobrepasa el estimado (lo cual llevaría a un esfuerzo restante negativo). El miembro del equipo asignado debería re-estimar

- **Actividad sin estimación**: la actividad no está estimada o su valor es 0. El miembro del equipo asignado debería estimar

Eventos que provocan una variación en el esfuerzo restante observado:

- **Cambios del esfuerzo invertido**: el esfuerzo invertido se ha modificado. Por ejemplo, se había registrado 10 horas de trabajo cuando realmente debían de ser 5 horas. También en casos en los cuales no se registra el esfuerzo en el momento, posteriormente es posible registrarlo
- **Incremento en la estimación**
- **Decremento en la estimación**
- **Introducción de estimación faltante**: indica que se ha estimado una actividad que el día anterior no tenía estimación
- **Actividad asignada a un miembro específico del equipo**: una nueva actividad de una WU ha sido asignada a un miembro. Esto es sólo relevante cuando se trata de la Gráfica Burn Down filtrada con un miembro del equipo específico
- **Actividad desasignada de un miembro específico del equipo**: una actividad de una WU ha sido desasignada de un miembro. Esto es sólo relevante cuando se trata de la Gráfica Burn Down filtrada con un miembro específico
- **WU nueva**: WU creada y añadida a la iteración
- **WU eliminada**
- **WU desestimada**. Su esfuerzo restante se considera igual a 0
- **WU añadida**: WU que se ha añadido a la iteración (ya existía sin iteración asignada o en otra iteración)
- **WU quitada**: WU que estaba el día anterior en la iteración y se ha cambiado de ésta

Para ilustrar el uso de estos eventos en la interpretación de la gráfica, a continuación comentamos un ejemplo. La Ilustración 9 muestra la Gráfica Burn Down

correspondiente a la actividad Programación de la versión 0.2 de un determinado producto. En la gráfica, el día 20 de Abril se observa un descenso del esfuerzo restante. Este descenso, a priori lo podemos asociar al descenso del esfuerzo estimado. Pero cuestiones como: *¿por qué ha descendido el esfuerzo estimado?, ¿se ha movido, eliminado o desestimado trabajo?, ¿algún miembro del equipo ha ajustado alguna estimación?*, no se pueden responder a simple vista. Para responder tales cuestiones es esencial la información de **la tabla *Daily Events***. En la Ilustración 10 vemos los eventos asociados a la actividad Programación que ocurrieron el día 20 de Abril; hubo un decremento en la estimación de la actividad Programación en 3 WUs y se han introducido 2 nuevas estimaciones que faltaban (en la Ilustración 9 vemos que el día 19 de Abril faltaba por estimar la actividad Programación en dichas WUs). De esta forma sabemos que el descenso del esfuerzo restante se debe a un decremento en la estimación de 3 WUs, aunque además se hayan incluido 2 nuevas estimaciones antes no consideradas.

Finalmente, el disponer de información detallada de lo realizado durante cada iteración, aporta información útil para **reuniones de revisión de la iteración** o **reuniones de retrospectiva**, pudiendo llegar a evaluar acciones de mejora en el proceso mediante la comparación de datos de diferentes iteraciones y su tendencia.

Capitulo 3. Mantenimiento de la especificación de requisitos

3.1. Test Driven Requirements Engineering

El resumen de Test Driven Requirements Egineering presentado en este capítulo se ha elaborado a partir del contenido de la Tesis de María Company Bria)[1]

El enfoque **Test-Driven Development** (TDD)[7] se basa en que las pruebas deben dirigir el desarrollo del producto software. El TDD se ha aplicado esencialmente en el ámbito de la implementación, particularmente siguiendo el planteamiento "no escribir código hasta disponer de las pruebas que debe satisfacer dicho código". El TDD ha tenido un fuerte impulso con las metodologías ágiles, las cuales lo incorporan entre sus prácticas esenciales.

En productos software industriales, cuando se utilizan prácticas de ingeniería de requisitos, éstas mayoritariamente están soportadas por lenguaje natural, lo cual conlleva los ya conocidos inconvenientes relativos a ambigüedad. Sin embargo, la necesidad de validación es más importante que las ventajas que puede ofrecer una especificación más formal y rigurosa de los requisitos. El cliente debe poder leer y comprender los requisitos para así poder dar su conformidad respecto de ellos. Las técnicas más populares para especificación de requisitos se resumen en **Casos de Uso** (utilizados en metodologías tradicionales, como RUP [8]) e **Historias de Usuario** (fichas de XP [9] o representaciones similares en otras metodologías ágiles como Scrum [10]). Si bien los elementos Actor (o tipo de usuario) y Requisitos (Caso de Uso o Historia de Usuario) pueden visualizarse y manipularse gráficamente o en fichas, la descripción de cada requisito se elabora esencialmente de forma textual en lenguaje natural.

Para definir los requisitos es clave involucrar al cliente. El acuerdo/contrato con el cliente y la planificación del desarrollo del producto deberían estar basados en los requisitos que se pretenden incorporar en el producto. Los requisitos son el objetivo a conseguir, es decir, es lo que espera el cliente que el producto software satisfaga.

El **modelo en V** considera las pruebas como un proceso que corre en paralelo con el análisis y el desarrollo, en lugar de constituir una fase aislada al final del proceso. En la

representación gráfica clásica del Modelo en V (véase Ilustración 11), las fases de desarrollo de software aparecen a la izquierda (desde requisitos hasta la implementación de unidades de código) y los correspondientes niveles de pruebas a la derecha (desde Pruebas Unitarias hasta Pruebas de Aceptación).

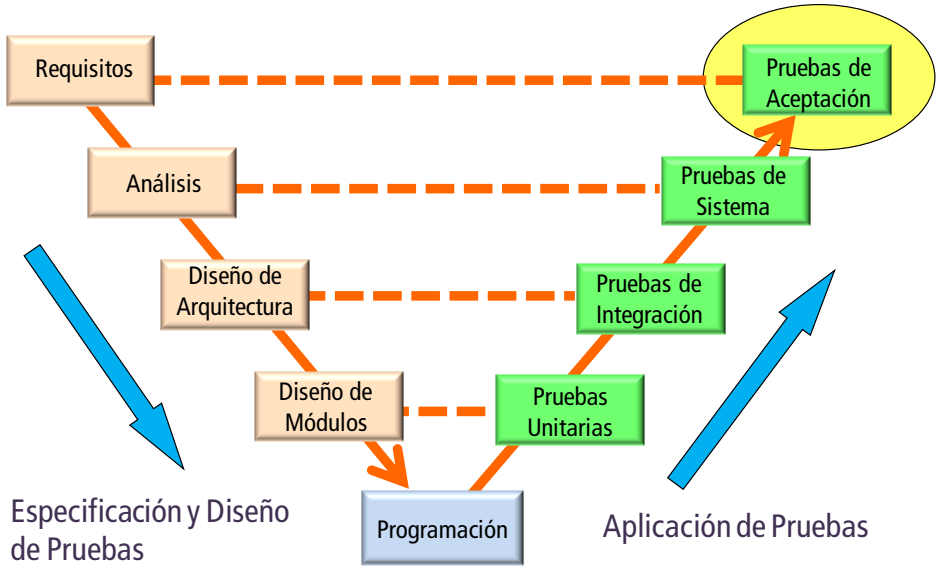


Ilustración 11 El Modelo en V

Partiendo de los requisitos o del diseño (a la izquierda) se deben planificar y preparar los niveles de pruebas correspondientes (a la derecha). En general, cada actividad de pruebas a la derecha valida la actividad enfrentada de la izquierda. También en general, los niveles superiores de pruebas en el diagrama están basados en **caja negra** (pruebas basadas en las especificaciones) y los niveles inferiores están basados en **caja blanca** (basadas en la estructura interna de los componentes del sistema).

La mayoría de las metodologías se basan en este modelo de pruebas, sin embargo, aquellas que apuestan decididamente por modelos de procesos iterativos e incrementales (RUP, entre las metodologías tradicionales, y todas las metodologías ágiles) utilizan de forma más explícita las **pruebas de sistema** y **pruebas de aceptación**. La única diferencia entre estos dos tipos de pruebas es el agente que las crea, las de aceptación son creadas por el cliente y las de sistema son creadas por el desarrollador. Aunque la mayoría de metodologías conocen e utilizan las **Pruebas de Aceptación**, en muy pocas las utilizan como hilo conductor del proceso de desarrollo software. En

estas metodologías utilizan las Pruebas de Aceptación como **criterio de éxito de cada iteración**. Esto nos llevaría a clasificar estas metodologías como Test-Driven, en un contexto más de planificación y desarrollo global. Sin embargo, en ellas no existe una analogía entre el Test-Driven del ámbito de implementación y lo que sería **Test-Driven en el ámbito de los requisitos y de la planificación**. El planteamiento en este último ámbito sería "*no incorporar un requisito a una iteración (o al plan en general) sin haber antes establecido sus Pruebas de Aceptación*".

En las metodologías populares (tradicionales o ágiles), los **artefactos Requisito y Prueba de Aceptación** son tratados de forma separada y abordados en diferentes actividades y momentos del desarrollo. Una vez establecidos los requisitos, se definen las Pruebas de Aceptación para ellos. Además, usualmente están involucrados dos roles también diferentes, Ingeniero de Requisitos (Analista o Analista Funcional) y Tester (es el tester especializado en Pruebas de Aceptación, no en Pruebas Unitarias y de Integración, las cuales normalmente son responsabilidad del programador).

Esta propuesta se denomina **Test-Driven Requirements Engineering (TDRE)** por referirse a TDD pero en el ámbito de los requisitos y la planificación. TDRE integra los artefactos, actividades y roles asociados a la especificación y validación de los Requisitos y de las Pruebas de Aceptación. El concepto de Requisito se convierte en un contenedor de Pruebas de Aceptación, y son éstas las que adquieren protagonismo como especificación de cada requisito. Una de las ocho "buenas características" que recomienda la **IEEE 830** [11] para una especificación de requisitos se refiere a que cada requisito debe ser "**Verificable**".

Si bien, como especificación de requisitos, las Pruebas de Aceptación pueden sufrir los mismos inconvenientes que los requisitos tradicionales (al usar lenguaje natural para especificarlas), una Prueba de Aceptación por su **granularidad y verificabilidad** es una unidad que determina de forma precisa una parte del comportamiento del sistema. Así, otras características recomendadas por el estándar IEEE 830 (Corrección, No ambigüedad, Completitud, Consistencia, Orden por importancia o estabilidad, Modificabilidad y Trazabilidad) también se ven favorecidas.

Aunque los requisitos y las actividades asociadas han ido haciéndose espacio entre los aspectos esenciales de un proyecto de desarrollo de software, para las pruebas no ha ocurrido lo mismo y sólo equipos de desarrollo con cierta madurez las integran como pieza imprescindible en sus procesos de desarrollo. Existen dificultades para la introducción de una "cultura", disciplina y prácticas de pruebas en un equipo de desarrollo. Entre los obstáculos o malas estrategias podemos destacar: carencia de un proceso de desarrollo que integre las actividades de pruebas con el resto de actividades, sobrevaloración de la automatización de las pruebas (y particularmente las Pruebas Unitarias) como objetivo inmediato o único, y el poco énfasis en "rentabilizar" el esfuerzo invertido en pruebas. Lo anterior hace que la iniciativa de incorporación de pruebas además de ralentizarse en cuanto a resultados, vaya perdiendo fuerza en su proceso de implantación en la organización.

En TDRE la estrategia es implantar una "cultura" de pruebas basada en el aprovechamiento de éstas como hilo conductor del proceso de desarrollo. TDRE está incorporado en la metodología **TUNE-UP** [2,3,4,5,6] y en su herramienta de apoyo **TUNE-UP Software Process**.

Para comprender cómo los requisitos pueden ser especificados mediante Pruebas de Aceptación comenzaremos con un ejemplo. Consideremos el requisito "Retirar dinero" en el contexto de un cajero automático. Una típica especificación narrativa podría ser la siguiente:

"El cliente debe poder retirar dinero del cajero en cantidades seleccionables. Siempre recibe un comprobante, a menos que el cajero se quede sin papel. Cuando se trata de un cliente preferencial puede retirar más dinero del que tiene en su cuenta, pero se le debe advertir que se le cobrarán intereses. El cliente debería poder cancelar en cualquier momento antes de confirmar el retiro de dinero. Las cantidades deberían poder servirse con los billetes que en ese momento tenga el cajero y no se deberían aceptar otros montos. Sería conveniente avisar cuando el cajero esté realizando operaciones internas mostrando un mensaje: El dinero retirado de la cuenta debe poder comprobarse en los registros de movimientos de la cuenta…".

La Ilustración 12 ilustra algunas alternativas de especificación para este requisito (incluyendo la anterior descripción narrativa como opción por defecto). Los iconos reflejan la conveniencia de cada alternativa de especificación.

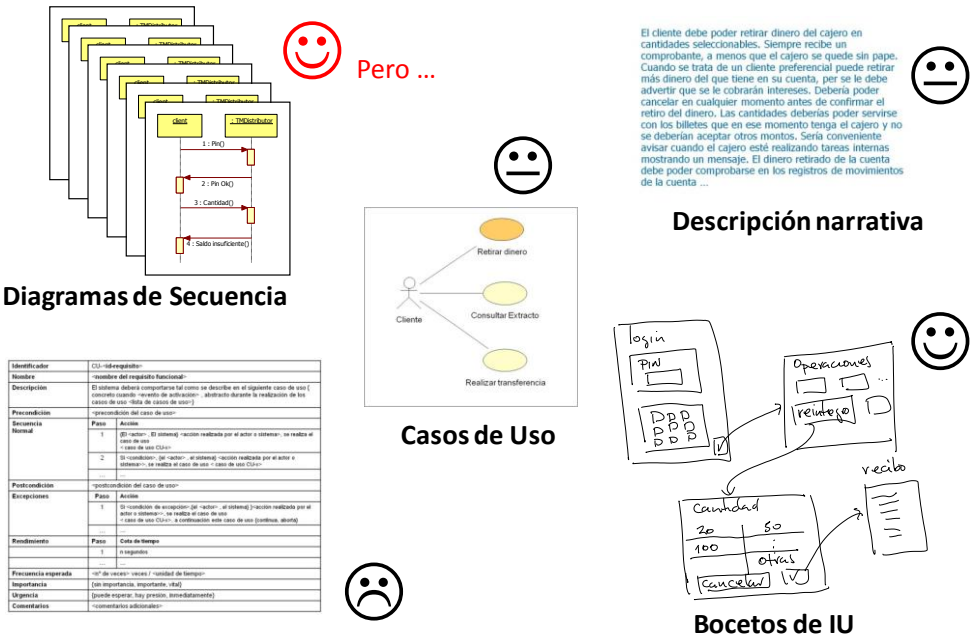


Ilustración 12. Alternativas populares para la especificación de requisitos

Elaborar un **Diagrama de Secuencia** para definir cada escenario de ejecución del requisito puede parecer interesante, sin embargo, en general no resulta apropiado por la gran cantidad de diagramas generados. Resulta más interesante la identificación de los escenarios que la ilustración de cada uno de ellos en un diagrama.

La **descripción narrativa** no es descartable, al menos para dar una breve definición del requisito centrándose en definir los conceptos involucrados (con la idea de un glosario o de un sencillo Modelo de Dominio).

Un **Modelo de Casos de Uso** no es mala idea, especialmente para organizar y visualizar los requisitos de un sistema nuevo. Sin embargo, un Modelo de Casos de Uso no es apropiado para ilustrar la estructura de requisitos detallada de un producto software en situaciones de mantenimiento a más largo plazo, ya que un producto software de tamaño medio puede tener miles de requisitos. La visualización y gestión de gran cantidad de requisitos necesita de mecanismos más apropiados.

Los **bocetos** (visualizaciones muy preliminares) de la Interfaz de Usuario (IU) son siempre bienvenidos pues son una herramienta efectiva de comunicación y validación con el cliente, el cual puede hacerse una idea del producto. En este contexto de requisitos no debe pretenderse realizar el diseño final de las IUs sino más bien paneles con cierto ámbito de datos, sin profundizar en tipos de controles de interfaz o cuestiones de estética de formularios/páginas.

Las **plantillas** son una de las alternativas de especificación más usadas para Casos de Uso. Las plantillas son elegantes y proporcionan una sensación de orden en la especificación. Sin embargo, en general resultan contraproducentes ya que tienden a dar un tratamiento uniforme en cuanto al nivel de detalle para todos los requisitos. En aquellos muy simples se tiende a incluir cosas obvias o irrelevantes sólo para poder cubrir todos los apartados de la plantilla. Cuando un requisito incluye varios (o muchos) escenarios, el intento por sintetizar todos los escenarios en una plantilla (que sólo ofrece pasos y excepciones) lleva normalmente a especificaciones enrevesadas.

TDRE apuesta por especificar los requisitos usando los elementos que se muestran en la Ilustración 13: una **breve descripción narrativa** que establece los conceptos y motivación del requisito, **bocetos de la IU** (si procede) y una **lista de Pruebas de Aceptación** (PAs). Estas PAs se refinarán desde simples frases que dan nombre a los escenarios, hasta pruebas diseñadas, listas para ser aplicadas o para automatizarse y aplicarse en regresión. Así, los requisitos actúan como contenedores para las PAs.

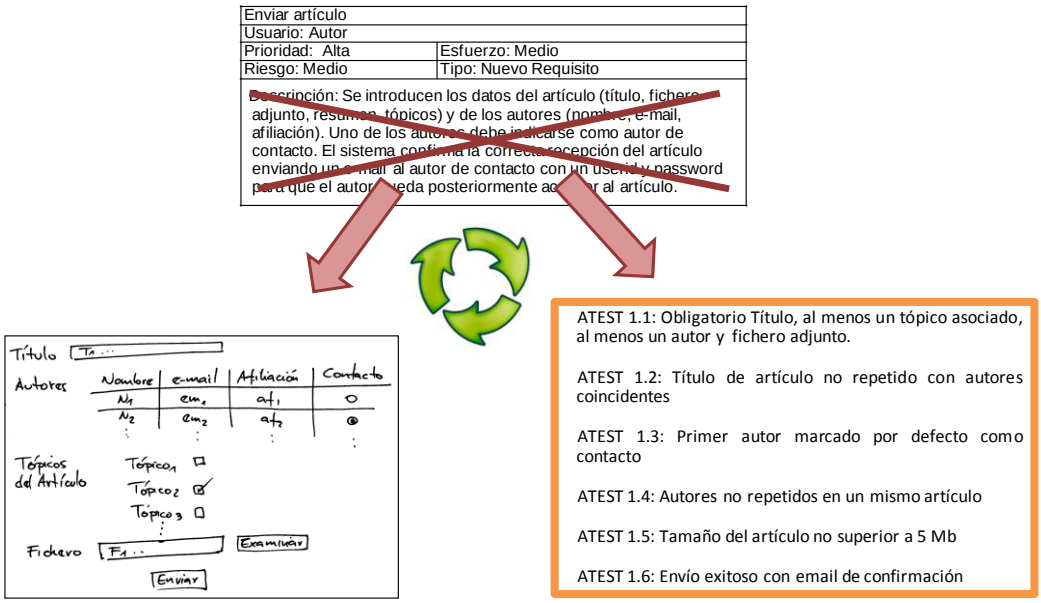


Ilustración 13. Especificación de requisitos en TDRE

Dependiendo del requisito, podría ser útil utilizar otras formas de especificación con carácter complementario, por ejemplo un **Diagrama de Actividad** si el comportamiento asociado al requisito es de carácter algorítmico o un **Diagrama de Estados** si el comportamiento incluye habilitación o deshabilitación de acciones de acuerdo con el estado del sistema. La premisa esencial es pragmatismo respecto de la especificación, con lo cual no se descarta el uso combinado de alternativas de especificación, pero el criterio primordial debe ser el rentabilizar el esfuerzo en especificación y facilitar el mantenimiento de dicha especificación. Por otra parte, desde el punto de vista de esfuerzo de mantenimiento, especialmente en cuanto a consistencia, es importante no abusar de solapes o duplicaciones de especificaciones en diferentes medios de representación.

Las miles de PAs que fácilmente puede llegar a tener un producto software deben organizarse adecuadamente para poder ser gestionadas de forma eficiente. Un **grafo dirigido** (Ilustración 14) es una representación adecuada para realizar un refinamiento por niveles. Dicho grafo permite tanto la visualización de relaciones de descomposición como de dependencia entre requisitos. Así, cada nodo es un requisito funcional o no funcional. Los arcos entre nodos establecen relaciones padres-hijos (mediante las cuales se hace una descomposición de los requisitos) o relaciones de dependencia del tipo “nodos que afectan nodos” (estas relaciones son clave para realizar análisis de

impacto). Los nodos de la parte más alta podrían, por ejemplo, seguir la clasificación de características ofrecida por la **ISO/IEC 9126** como punto de partida para la definición de los requisitos del sistema.

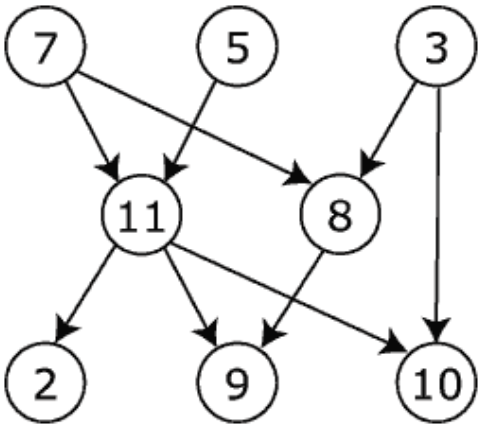


Ilustración 14. Representación de un grafo acíclico dirigido

La PA cuando va pasando por el proceso de desarrollo software va cambiando de estados, dependiendo de los recursos que disponga la empresa tendremos más o menos agentes interactuando con las PAs a lo largo del proceso. La situación ideal sería tener al menos tres roles diferentes: el **Ingeniero de Requisitos** (a nivel de empresa conocido como Analista funcional o Analista de Negocio), el **Programador** y el **Tester**. Si por falta de recursos no se tuvieran los tres roles, el mismo agente que analiza podría programar o el mismo que analiza podría testear, lo conveniente es que no testee el mismo agente que ha implementado.

Los estados por los que pasa una PA se explican a continuación:

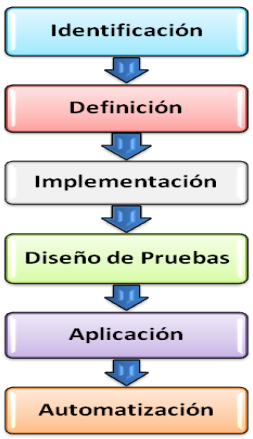


Ilustración 15 Estados de un PA

- **Identificación:** Se capturan los requisitos negociando con el cliente y se identifica el nombre de las PAs. Cuando se tienen las PAs identificadas ya se pueden tomar decisiones de planificación.

- **Definición:** Se especifica el contenido de la PA y se hace el análisis de impacto (a que otras partes del producto podría afectar el cambio). Una vez evaluamos el análisis de impacto y

se valida con el cliente podemos decidir si el cambio que se define en la prueba es aceptado o no.

- **Implementación:** Durante la implementación el programador gracias a las PAs (se podrían considerar como un contrato Analista-Programador) tiene un criterio de éxito medible ya que escribe código con la idea de satisfacer la PA. Una vez ya ha pasado todas las PAs ha terminado de implementar.

- **Diseño de Pruebas:** El tester es el encargado de buscar las diferentes combinaciones de datos adecuadas para las diferentes instanciaciones de la PA.

- **Aplicación:** Se aplica la PA y se valida la implementación.

- **Automatización:** Hasta este nivel todo es manual, a partir de aquí si la empresa tiene suficientes recursos y se va a incluir la automatización de la prueba se podrían aplicar pruebas de regresión automatizadas

A lo largo de este trabajo nos centraremos más detalladamente en la identificación y definición de las PAs, lo cual está más asociado al marco de trabajo del analista. El diseño y ejecución de las PAs es responsabilidad del tester y no lo veremos en detalle.

Siguiendo con el ejemplo anterior, el requisito "**Retirar dinero**" podría ser un nodo de la estructura de requisitos. Los nombres que identifican sus PAs podrían ser:

1. *Reintegro usando cantidades predefinidas habilitadas*
2. *Reintegro con cantidad introducida por cliente*
3. *Reintegro saldo < cantidad*
4. *Cancelación de operación*
5. *No disponibilidad de billetes*
6. *No disponibilidad de papel para recibo*
7. *Reintegro saldo < cantidad con cliente preferencial*
8. *Excedido tiempo de comunicación con sistema central*
9. *Aviso de operaciones internas del cajero*
10. *Excedido tiempo de espera para introducción de acción*

Una PA tiene como propósito demostrar al cliente el cumplimiento parcial o total de un requisito del software. Las características de una PA son:

- Una PA describe un escenario (secuencia de pasos) de ejecución o un uso del sistema desde la perspectiva del cliente. Las PAs cubren desde escenarios típicos/frecuentes hasta los más excepcionales
- Puede estar asociada a un requisito funcional o requisito no funcional. Un requisito tiene una o más PAs asociadas
- Una PA puede tener infinitas instanciaciones (ejecuciones con valores concretos)

La definición de una PA se separa en cuatro apartados: Condición, Pasos, Resultado Esperado y Observaciones.

- **Condición.** Es opcional, y se utiliza para establecer condiciones previas antes de aplicar los pasos de la prueba. Normalmente se refieren al estado de la BD antes de ejecutar la prueba y/o la navegación necesaria en la IU para localizarse en el punto adecuado para realizar la prueba (cuando no sea obvio)
- **Pasos.** Son las acciones de interacción del actor con el sistema. Cuando son varias acciones, éstas pueden ponerse en una lista numerada. Deben ser simples, del estilo "seleccionar…", "introducir...", evitando hacer referencia explícita a controles de interfaz o cualquier tecnicismo que dificulte su validación con el usuario
- **Resultado esperado**. Es el efecto de las acciones de interacción del actor. Cada acción puede provocar uno o más resultados. Es importante que cuando se trate de mensajes al usuario se incluya el texto como parte del resultado esperado, así el programador dispone de dicha información ya validada con el cliente

- **Observaciones.** Es un apartado opcional, son recomendaciones que el analista estima conveniente hacerle al tester y/o programador

Las condiciones y/o resultados esperados pueden establecerse en el ámbito del requisito contenedor de la PA o de otros requisitos. Esto, como indicaremos más adelante, permitirá establecer relaciones de dependencia entre requisitos.

A continuación se presenta como ejemplo la definición de la PA "*Reintegro saldo < cantidad*".

CONDICIÓN

- ➢ Debe existir un cliente normal (esta característica se establece en el apartado datos básicos del cliente)

PASOS

- → Intentar reintegro con cliente normal y con cantidad solicitada mayor al saldo

RESULTADO ESPERADO

- ✓ Se muestra el mensaje "La cantidad solicitada supera su saldo actual, vuelva a introducir la cantidad" y se retorna a la ventana para introducir la cantidad

Es importante que el analista establezca un orden en las pruebas, desde las correspondientes a escenarios más típicos/frecuentes hasta aquellos más excepcionales. Éste orden servirá de guía de implementación a los programadores y al trabajo del tester. El programador debería escribir el código con la idea de satisfacer las PAs de forma incremental, hasta implementar toda la funcionalidad y superar todas las PAs. Por su parte, y adicionalmente, el programador podría realizar testeo unitario y de integración para las piezas de código que escriba. Dichas Pruebas Unitarias y de Integración, no siempre están relacionadas directamente con los requisitos puesto que su propósito se orienta más a verificación del software que a su validación (las PAs validan la implementación respecto de los requisitos del cliente). Sin embargo, en casos en los cuales las Pruebas Unitarias y de Integración se derivan o tienen una relación más directa con los requisitos, éstas se ven favorecidas porque en TDRE los requisitos están especificados como pruebas. Por otra parte, el tester diseñará, aplicará, y documentará una o más ejecuciones instanciadas (con valores concretos) para cada PA.

Las PAs se rentabilizan durante el proceso de desarrollo de software porque permiten:

- Especificar y validar los requisitos.
- Valorar adecuadamente el esfuerzo asociado a la incorporación de un requisito. Es más sencillo valorar el esfuerzo que requiere la satisfacción de cada PA.
- Negociar con el cliente el alcance del sistema en cada iteración de desarrollo. Las PAs introducen un nivel de granularidad útil para negociar el alcance con el cliente. Un requisito no necesita implementarse "todo o nada", puede hacerse de forma incremental postergando la satisfacción de ciertas PAs.
- Guiar a los desarrolladores en la implementación ordenada del comportamiento asociado a un requisito. Si bien las PAs no son tan exhaustivas como las Pruebas Unitarias generan gran parte de la cobertura de código. Esto no significa necesariamente prescindir del resto de niveles de pruebas del Modelo V, sino que en un contexto de recursos limitados, las PAs deberían ser las pruebas esenciales o mínimas que deberían definirse y aplicarse al sistema.
- Identificar oportunidades de reutilización. El detalle y precisión proporcionado por las PAs permite identificar en la especificación de requisitos posibles solapes de comportamiento que den origen a oportunidades de su reutilización (lógica de la aplicación).

3.2. TDRE en TUNE-UP

TUNE-UP presta atención tanto al desarrollo inicial del producto como a su mantenimiento (**"Todo producto exitoso necesitará de mantenimiento**"). Los requisitos en TUNE-UP son especificados mediante PAs. En TUNE-UP el cliente con la ayuda del analista es el encargado de definir las WUs en términos de PAs. Posteriormente el programador escribe código para satisfacer las PAs, y por último, el Tester establece combinaciones de datos para cada una de las PAs y las aplica.

En TUNE-UP la estructura de requisitos se representa como un grafo acíclico dirigido cuyos nodos son contenedores de PAs. Como se muestra en la Ilustración 16 un

cambio en el comportamiento del producto viene dado por una WU, la cual afecta a uno o más nodos de la estructura de requisitos del producto, añadiendo, modificando o eliminando PAs.

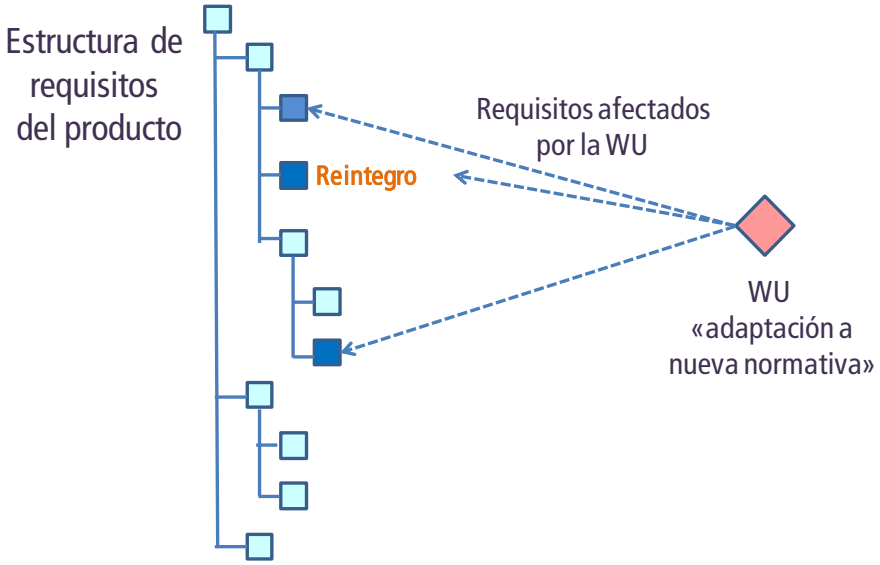


Ilustración 16. Estructura de Requisitos del producto

A continuación ilustraremos cómo TDRE se ha integrado en TUNE-UP Software Process mediante el módulo específico denominado REM.

3.2.1 Proceso dirigido por las pruebas de Aceptación

El primer paso a realizar cuando el equipo de desarrollo o el cliente quieren hacer algún cambio de comportamiento dentro del producto es crear la WU correspondiente. Las WUs se pueden crear desde diferentes sitios en TUNE-UP (PEP, WUM y REM). El contenido del REM es muy similar al de la **pestaña Product Change Scope** dentro del WUM, la única diferencia es que el REM está orientado a consultar el comportamiento pasado, actual y previsto del producto sin estar en el contexto de una unidad de trabajo determinada. Esta información es muy útil en el momento de proponer un nuevo cambio en el producto para por ejemplo, evitar solapes e inconsistencias entre cambios o programar de forma racional futuros cambios del producto prestando atención al conjunto de cambios pendientes de implementar.

El REM tiene dos modos de trabajo: **Standard Mode**, el cual es el modo por defecto al acceder al REM, y el **New WU Preparation Mode**, el modo en el cual se permite crear una nueva WU directamente desde el REM. En este último modo se puede hacer todo lo ofrecido en el modo estándar, pero adicionalmente, se muestran checks en los nodos del grafo para marcar los nodos afectados por la WU que posteriormente se creará. Así, resulta integrada la tarea de verificar el comportamiento ya definido o por definir en los nodos en los cuales se pretende llevar a cabo un cambio, con respecto del hecho de indicar los nodos que se verán afectados.

Con el botón **New WU** se abrirá el formulario de creación de una WU (formulario que está sobrepuesto en la Ilustración 17). La WU creada desde el REM mantendrá las marcas que se hagan en el REM, pudiendo posteriormente el analista refinarlas según se requiera. El trabajo de definir detalladamente el cambio mediante PAs se hará posteriormente por el Analista.

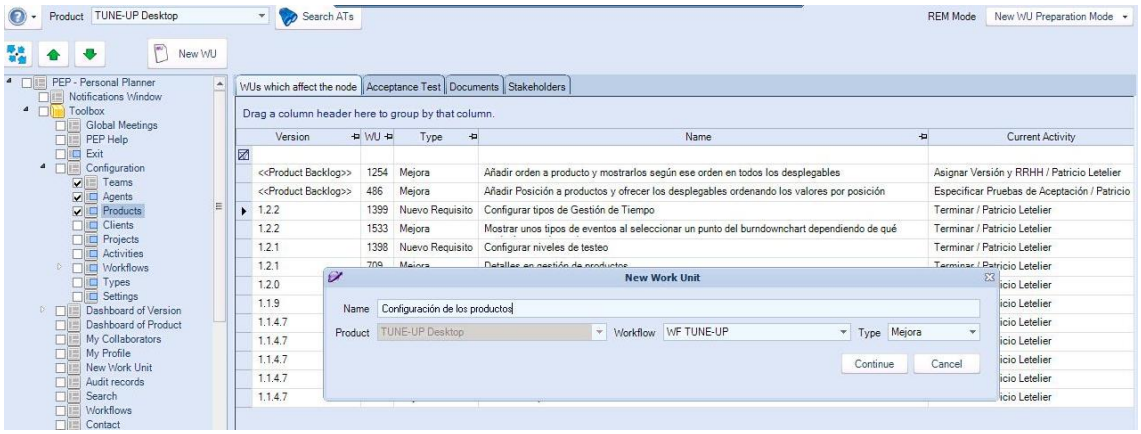


Ilustración 17. Creación de una WU desde el REM

En TUNE-UP el módulo WUM apoya a los agentes en sus actividades sobre una determinada unidad de trabajo. Desde este módulo y para cada unidad de trabajo, se puede acceder a la **pestaña Product Change Scope** (ver Ilustración 17), donde el analista revisaría las marcas de los nodos que se verán afectados y si es necesario tendría que modificarlas (podría darse el caso que el agente que ha creado la WU no conociera suficientemente la estructura de requisitos y se haya dejado nodos por marcar o haya puesto las marcas en niveles superiores). Una vez las marcas ya sean las correctas el analista se encargaría de definir los cambios en las PAs de dichos nodos.

Para el equipo de desarrollo es muy importante conocer qué nodos afecta cada unidad de trabajo. Esta información permite detectar posibles conflictos o solapes entre las WUs contenidas en la misma versión o entre WUs de diferentes versiones. Por ejemplo, si un analista está definiendo un cambio de comportamiento en un nodo, le interesará conocer cómo ha evolucionado su comporta-miento (conocer las WUs realizadas en el nodo con sus correspondientes cambios en PAs), el comportamiento actual del nodo (las PAs vigentes en el nodo) y los cambios pendientes en el nodo (WUs pendientes que afectarán al nodo, con sus correspondiente cambios propuestos para sus PAs).

En el lateral izquierdo de la Ilustración 18 se observa la estructura de requisitos del producto (desplegada de forma parcial), donde se ven marcados los nodos afectados en la WU. Los nodos de la estructura de requisitos pueden ser de diferente tipo y se diferencian por el icono que tienen delante: (Nodo que representa un formulario o una página), (Nodo que representa un elemento) y (Nodo abstracto).

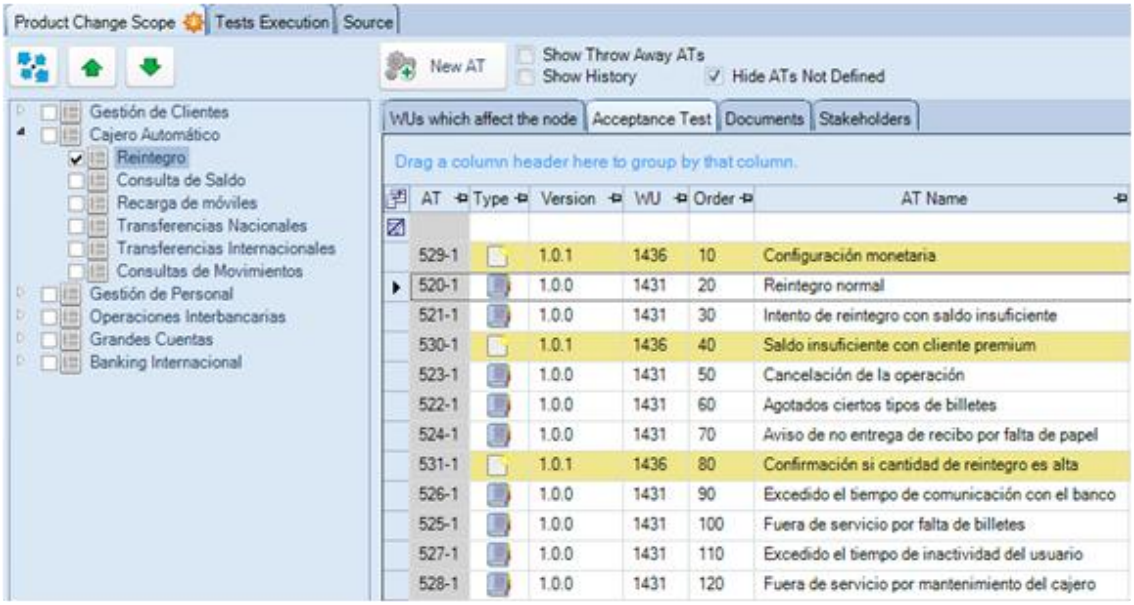


Ilustración 18. Pestaña Product Change Scope

En la **pestaña Acceptance Test** de la Ilustración 18 se observa la lista de PAs del nodo seleccionado (Nodo Usuario) en la estructura de requisitos. En este listado se muestran las PAs definidas o modificadas en esta WUM (marcadas en amarillo), aquellas que se implementaran en otras WUS y las que representan el comportamiento actual del nodo. También en cualquier momento si marcamos el check ***"Show History"*** se

mostrarán todas las PAs del nodo seleccionado, incluyendo aquellas versiones anteriores de una PA (pruebas que representen comportamiento pasado y están representadas con el icono).

Las consecuencias de la implantación de una unidad de trabajo se refleja claramente por los cambios en las pruebas del nodo: **PA nueva** (), **PA eliminada** () y **PA modificada** (). Además, las PAs que tienen el icono son **pruebas actuales** del nodo e indican comportamiento ya existente que se mantiene. Las pruebas existentes pueden marcarse por el analista o por otros agentes que participan en el proceso para que sean aplicadas como pruebas de regresión, de esta forma se asegura que el cambio no afecta dicho comportamiento. Esto último es útil para poder acotar el esfuerzo o tiempo requerido para aplicar las pruebas de regresión, pues incluso aunque dichas pruebas estuviesen automatizadas, no siempre es factible aplicarlas todas en el momento específico que lo requiere (por ejemplo, cada vez que el programador publica sus cambios de código, termina la implementación de una unidad de trabajo, cuando se termina una versión, etc.).

Los checks ***"Hide Ats Not Defined"*** y "***Show Throw Away ATs"*** (parte superior de la Ilustración 18) permiten filtrar las PAs mostradas en la pestaña Acceptance Tests. La **casilla Hide ATs Not Defined** oculta las **PAs ficticias**, las cuales se muestran para destacar que en ciertas WUs no se han definido PAs, con lo cual el cambio asociado no está explícito ni completo en las PAs existentes del nodo seleccionado. Serán WUs que al crearse han marcado los nodos que se van a ver afectados, pero aún no ha llegado a la actividad Analizar WU y no se ha especificado el cambio mediante PAs. Las pruebas Ficticias se diferencian por el icono y porque tienen como nombre "ATs NOT DEFINED". La **casilla Show Throw Away ATs** muestra en el grid las **PAs Throw Away,** las cuales expresan comportamiento que no debe presentarse, están asociadas a correcciones de fallos y no representan el comportamiento actual del nodo (Ejemplo Ilustración 19).

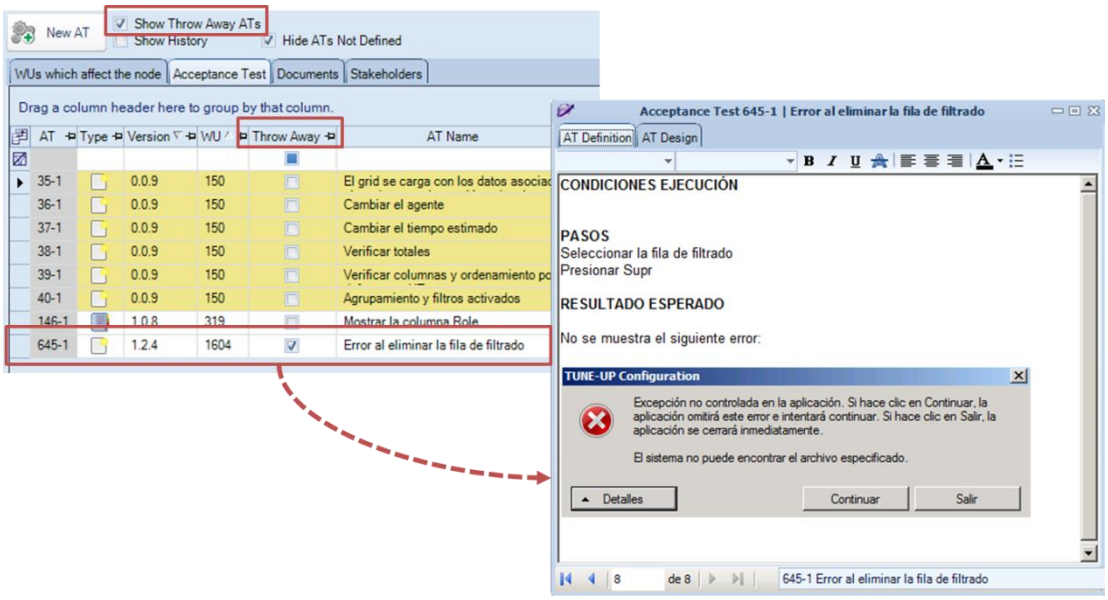


Ilustración 19. Ejemplo Prueba Throw Away

Para ver los detalles de una prueba al hacer doble click sobre el listado de PAs se accede al formulario de la prueba que se muestra en la Ilustración 20.

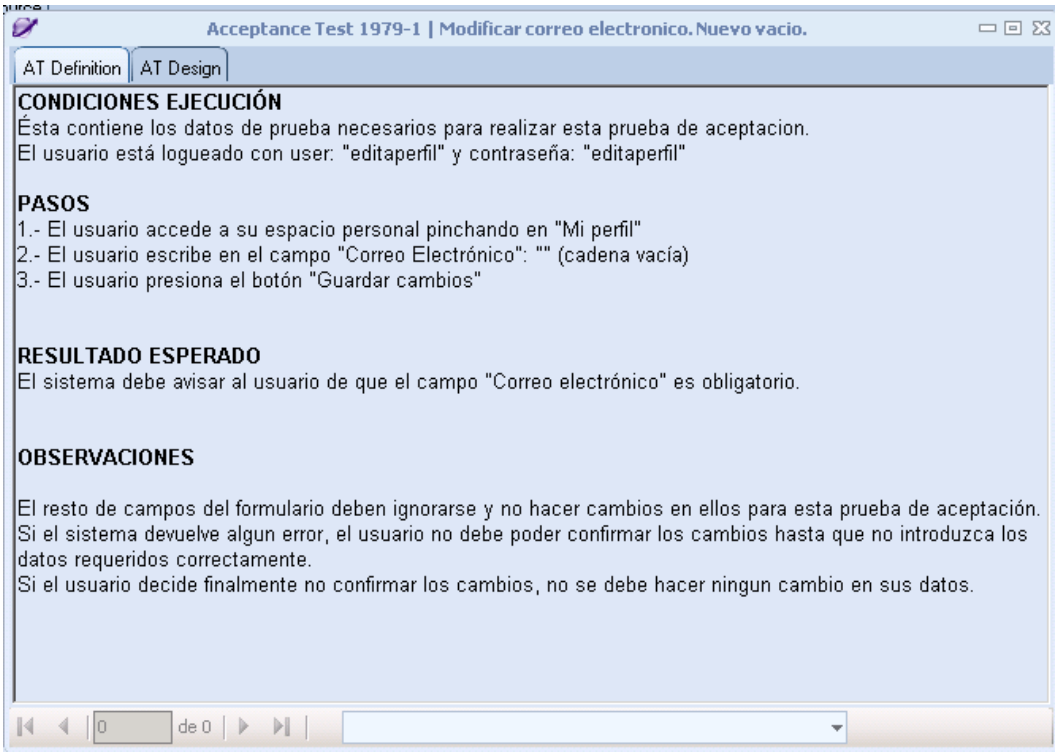


Ilustración 20. Formulario de Acceptance Test

En la pestaña **WUs which affect the node** de la Ilustración 18 se pueden observar las WUs que han afectado, están afectando o afectarán al nodo seleccionado en la estructura del producto. Para cada WU se muestra la versión en la que se incluyó la WU, código, tipo, nombre, actividad actual en la que se encuentra dentro del workflow y los agentes asignados.

También se dispone de otra pestaña **Documents** como espacio de documentos asociados al nodo, por ejemplo, modelos u otro tipo de especificación complementaria que se desee almacenar. Finalmente, es posible en la pestaña **Stakeholders** indicar los tipos de usuario u otros interesados en los servicios que ofrecerá dicha parte de la aplicación.

TUNE-UP permite además explotar la información de las PAs en el contexto de una unidad de trabajo desde la perspectiva del resto de los agentes que colaboran en su realización. TUNE-UP permite configurar los niveles de testeo que van aplicar las PAs, esto normalmente dependerá de los recursos de la empresa, en los siguientes ejemplos vamos a suponer que sólo disponemos de dos niveles de testeo: Programación y Testeo. Cada nivel tiene asociada información de seguimiento de las PAs, así cada participante (programador, tester) puede registrar sus ejecuciones de cada PA.

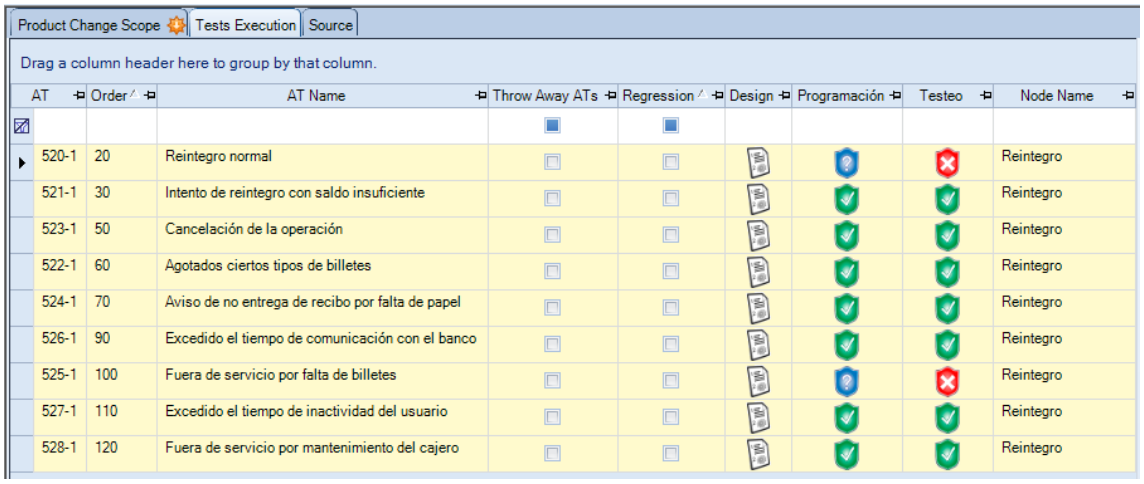
En la **pestaña Test Execution del WUM** (Ilustración 21) se muestra el listado de PAs definidas, modificadas o marcadas de regresión que el analista ha definido. En el grid se destaca el estado de aplicación de la PA, registrado por los agentes en los diferentes niveles de testeo.

Dichos estados de aplicación de la PA son:

- ***Vacío*:** la prueba no se ha aplicado
- : la última aplicación de la prueba ha sido exitosa
- : la última aplicación de la prueba ha detectado defectos
- **:** la prueba ha sido aplicada con éxito pero está marcada como pendiente de volver a aplicar, por ejemplo porque se ha modificado la PA o porque se

sospecha que otra prueba posteriormente implementada podría haberla afectado

- : la última aplicación de la prueba ha sido exitosa pero existe una aplicación en un nivel de testeo posterior (aplicación realizada por otro agente) en la cual se han detectado defectos, con lo cual una vez resueltos, debería volverse a aplicar la PA



AT	Order	AT Name	Throw Away ATs	Regression	Design	Programación	Testeo	Node Name
520-1	20	Reintegro normal						Reintegro
521-1	30	Intento de reintegro con saldo insuficiente						Reintegro
523-1	50	Cancelación de la operación						Reintegro
522-1	60	Agotados ciertos tipos de billetes						Reintegro
524-1	70	Aviso de no entrega de recibo por falta de papel						Reintegro
526-1	90	Excedido el tiempo de comunicación con el banco						Reintegro
525-1	100	Fuera de servicio por falta de billetes						Reintegro
527-1	110	Excedido el tiempo de inactividad del usuario						Reintegro
528-1	120	Fuera de servicio por mantenimiento del cajero						Reintegro

Ilustración 21. Pestaña Test Execution

Cada nivel de testeo crea sus propios registros de aplicación de pruebas constituyendo un histórico de las ejecuciones de una PA (parte inferior de la Ilustración 22). Por ejemplo, cuando un programador implementa y aplica una PA, registra un seguimiento OK para indicar que se ha superado con éxito dicha PA. Un seguimiento está formado por la WU, Fecha de ejecución, Agente, Nivel de testeo, el Resultado (OK, KO y OK!), un Fichero explicativo sobre el resultado (conteniendo instrucciones para la reproducción de los defectos detectados) y un apartado para añadir comentarios.

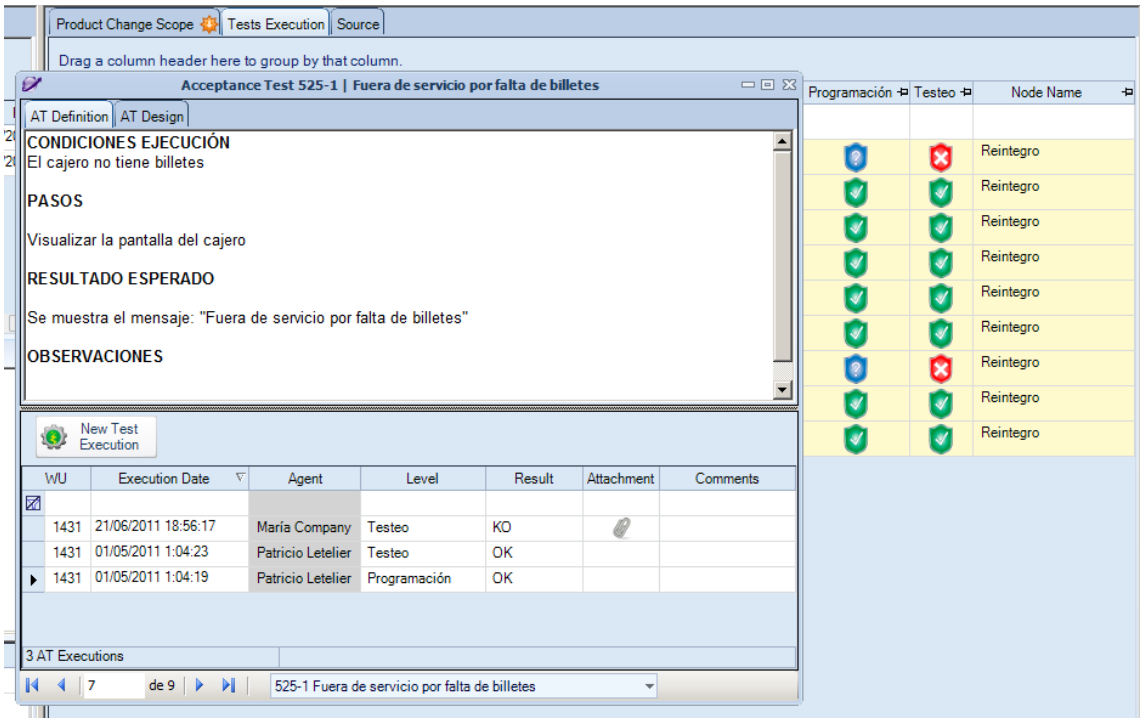


Ilustración 22. Ejecución de la PA

3.2.2 Pruebas de Aceptación y referencias a entidades

Una PA tiene como propósito demostrar al cliente el cumplimiento parcial o total de un requisito del software. Una PA describe un escenario de ejecución o de uso del sistema desde la perspectiva del cliente. Un requisito puede contener una o más PAs y éstas pueden estar asociadas tanto a requisitos funcionales como a no funcionales. A continuación se presenta como ejemplo la definición de la PA "Intento de reintegro con saldo insuficiente" (se trata del contexto de la funcionalidad Reintegro en un cajero automático).

CONDICIÓN
Cliente del tipo normal
Cliente con saldo positivo
Acceder a ventana de reintegro
PASOS
Introducir cantidad mayor que el saldo
RESULTADO
Se muestra el mensaje "Saldo insuficiente"
Se ofrece nueva introducción

En TUNE-UP la definición de una PA se compone de tres apartados: Condiciones, Pasos y Resultado. El cuerpo de cada apartado se escribe en lenguaje natural pero referenciando entidades del modelo de dominio del producto. Esto nos permite establecer automáticamente relaciones entre PAs que tienen referencia a las mismas entidades. Una entidad puede contener atributos o relaciones con otras entidades, el modelo de dominio que vamos a utilizar en este trabajo se muestra en la Ilustración 23

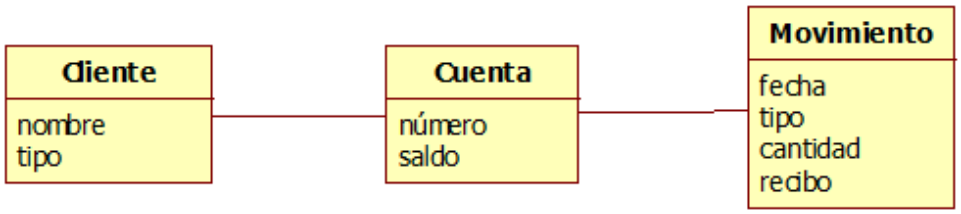


Ilustración 23. Modelo de dominio de ejemplo

En la Ilustración 24 se muestra un diagrama que ilustra las relaciones entre el atributo Saldo de la entidad Cuenta y las PAs de los nodos Reintegro, Gestión de Clientes y Transferencia. La relación entre una PA y una entidad surge cuando en la especificación de dicha PA, en alguno de sus tres apartados, se ha referenciado dicha entidad.

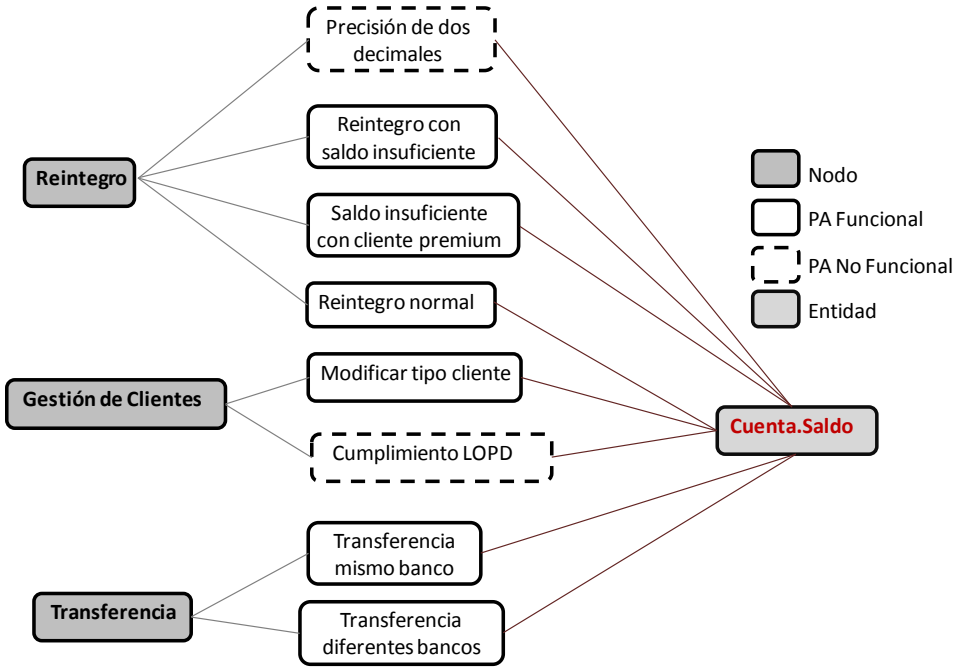


Ilustración 24 Representación relaciones PAs, Nodos y Entidades

3.2.3 Seguimiento de una iteración con TUNE-UP mediante PAs

En TUNE-UP, para realizar la planificación y seguimiento de las iteraciones, el jefe de proyecto utiliza el **módulo Version Content & Tracking**. La Ilustración 25 muestra parte de la **interfaz llamada "Version Content",** la cual contiene la lista de WUs de la versión de un producto. El jefe de proyecto puede consultar los datos resumidos de cada unidad de trabajo en la versión, en particular, puede conocer la actividad actual en que se encuentra dentro del workflow, su orden dentro de la versión, el esfuerzo que implica elaborar la unidad de trabajo, el agente asignado a las actividades principales del workflow, etc. Además, por cada nivel de testeo existirá una columna que refleja el estado de aplicación de las PAs de cada unidad de trabajo, mostrando el porcentaje obtenido a partir del número de pruebas con estado de aplicación OK dividido por el total de PAs. Con toda esta información el jefe de proyecto tiene una visión más detallada del estado de avance de la unidad de trabajo. Esta vista es muy reveladora, lo normal es que los porcentajes de cada nivel se vayan completando de forma secuencial, puesto que la unidad de trabajo va pasando por diferentes actividades

realizadas por agentes diferentes. Sin embargo, cuando aparecen las inevitables situaciones de re-trabajo, éstas quedan en evidencia por la forma incompleta que presentan los porcentajes de pruebas OK en cada nivel de testeo. La línea subrayada en la Ilustración 25 es un caso frecuente de retrabajo generado por saltos atrás en el proceso, en este caso en concreto el tester ha empezado a aplicar las PAs de la WU pero ha detectado fallos (ha marcado una prueba con KO) y por lo tanto le ha devuelto al programador la WU a la actividad Diseñar e Implementar para que solucione la prueba que tiene marcadas con interrogante.

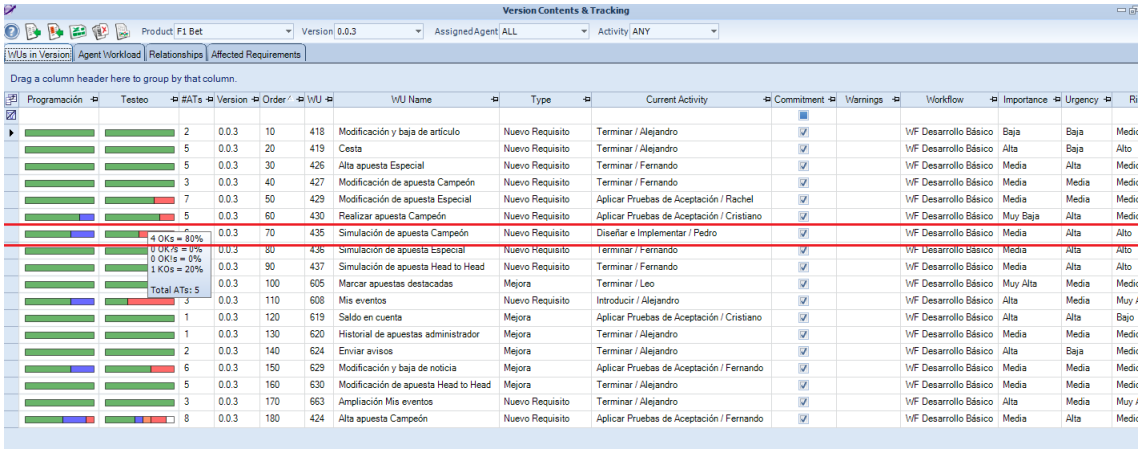


Ilustración 25. Pestaña Wus in Version del VCT

Para el jefe de proyecto no basta con conocer la actividad actual de la unidad de trabajo, puesto que además se puede estar realizando trabajo en paralelo respecto de la corrección de defectos detectados. En la práctica, sólo cuando los defectos son muy graves se devuelve a una actividad previa la unidad de trabajo. Por lo general los defectos se informan a los responsables de su corrección mediante mensajes, pero se continúa con el trabajo de la actividad donde ha llegado la unidad de trabajo.

El jefe de proyecto también puede disponer del estado de avance de las PAs de una versión con la **gráfica de estado de PAs** que se encuentra en el Dashboard. En la gráfica de la Ilustración 26 la línea vertical representa el número de PAs de una WU y la línea horizontal tiene una barra para cada nivel de testeo de una WU. En el ejemplo de la Ilustración sólo tenemos dos niveles de testeo, por lo tanto cada WU tiene representado el estado de las PAs en los niveles de programación y testeo. Esta vista ofrece una visión en forma de gráfica y por lo tanto más fácil de interpretar que las

columnas Programación y Testeo de la Ilustración 25. La única diferencia es que en VCT sólo se puede ver el estado de PAs de WUs de una versión determinada y en la gráfica de estado de PAs se pueden configurar, a petición del usuario a través de filtros, las WUs que se quieren mostrar en la gráfica.

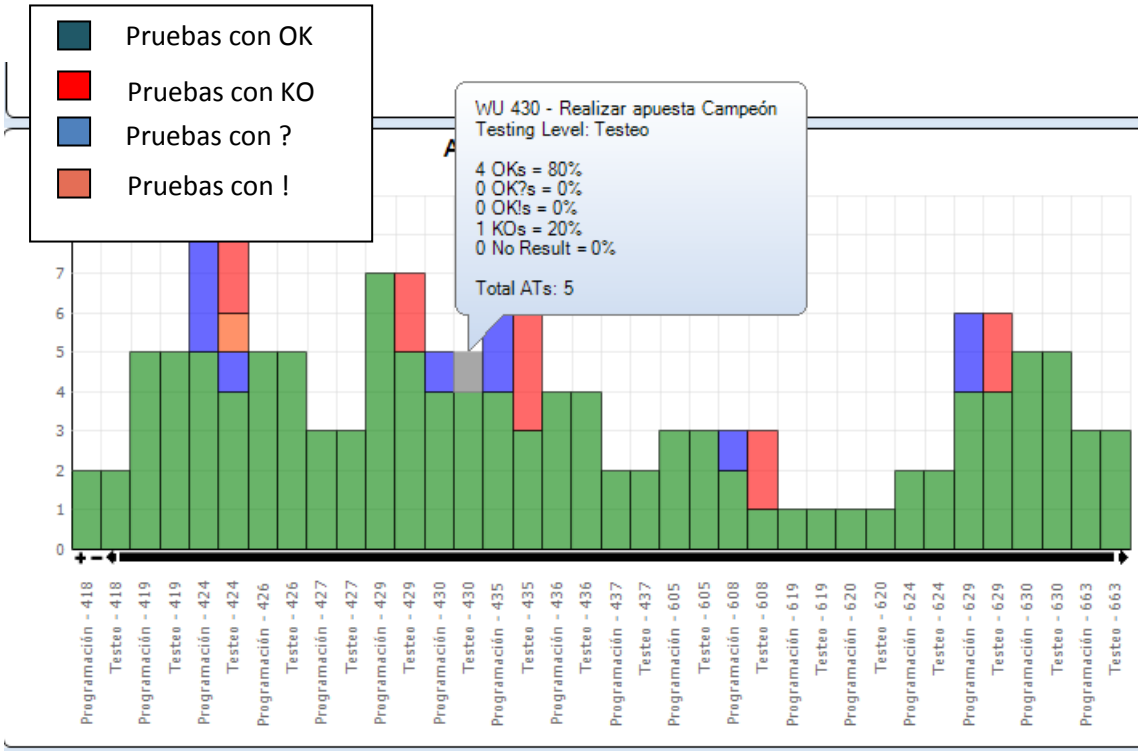


Ilustración 26. Gráfica de Estado de PAs

La pestaña **Affected Requirements** (Ilustración 27) es de gran utilidad para el jefe de proyecto ya que permite detectar posibles conflictos o solapes entre las WUs contenidas en la misma iteración o entre WUs de diferentes iteraciones. En la estructura del producto aparecen seleccionados los nodos que se ven afectados en WUs de la versión y entre paréntesis delante del nombre del nodo el número de WUs de la versión que modifican su comportamiento. Al activar un nodo de la estructura de requisitos (panel de la izquierda), en el panel de la derecha se cargan las WUs que afectan a dicho nodo en la versión (WUs con fondo oscuro) y en otras versiones (WUs con fondo blanco). Esta información ofrece las siguientes ventajas:

- **Detectar conflictos entre WUs.** Hay otra WU que afecta a la misma parte del producto que la WU de la versión y sería conveniente que se hiciera en la misma versión o incluso antes.

- **Detectar solapes entre WUs.** Nos ofrece una ayuda para evitar que hayan WUs duplicadas.

- **Asignar los mismos agentes.** Asignar a las WUs los mismos agentes que están asignados a las otras WUs que afectan a la misma parte del producto (columna asignados), ya que conocen mejor esa parte del producto y, además si están en la misma versión las WUs se evitan problemas de protección de código.

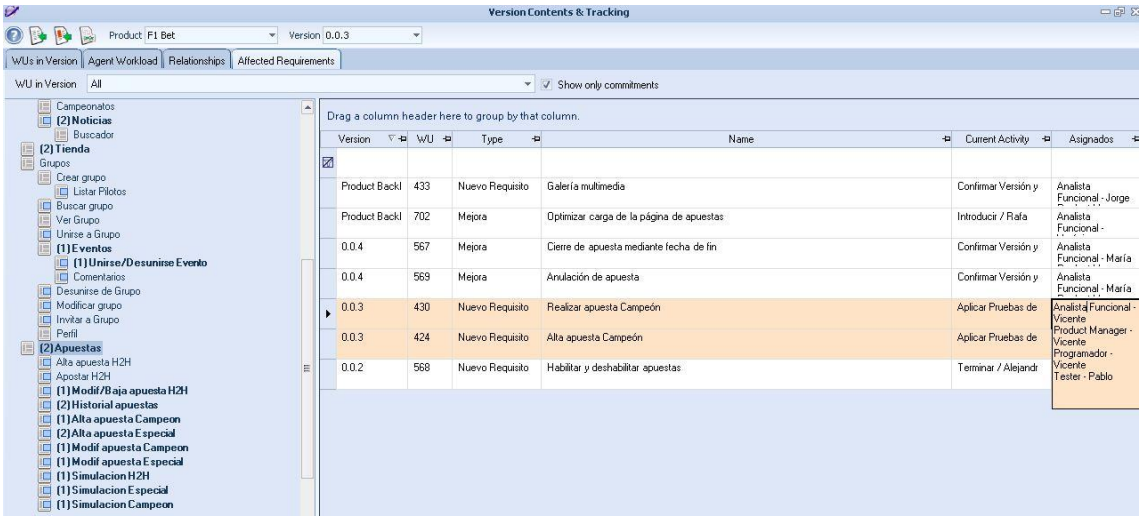


Ilustración 27. Pestaña Affected Requirements de VCT

3.3. Mantenimiento de PAs

3.3.1 Motivación

Durante el paso del tiempo dentro de un producto, se van creado nuevas WUs a las que los analistas van añadiendo nuevo comportamiento dentro del sistema que luego los desarrolladores irán implementando, estas pruebas de aceptación que diseñan los analistas para que sean implementadas por los programadores, van convirtiéndose en pruebas actuales que representan el comportamiento del sistema.

A veces, surgen una serie de problemas durante este proceso, por ejemplo, los analista no diseñan todas la pruebas y por lo tanto existen carencias en la

definición del comportamiento del sistema a través de las pruebas actuales, es decir, existe el comportamiento en el producto pero no se había creado la prueba de aceptación correspondiente que especifica dicho comportamiento y por lo tanto no existe la prueba actual. En otros casos el problema es menos grave, existe la prueba actual pero no está correctamente diseñada o incluso falta algún comportamiento en su descripción.

En la Ilustración 28 se puede observar como la prueba de la izquierda requiere una refactorización para añadir funcionalidad no definida mientras que la prueba de la derecha es un prueba completa.

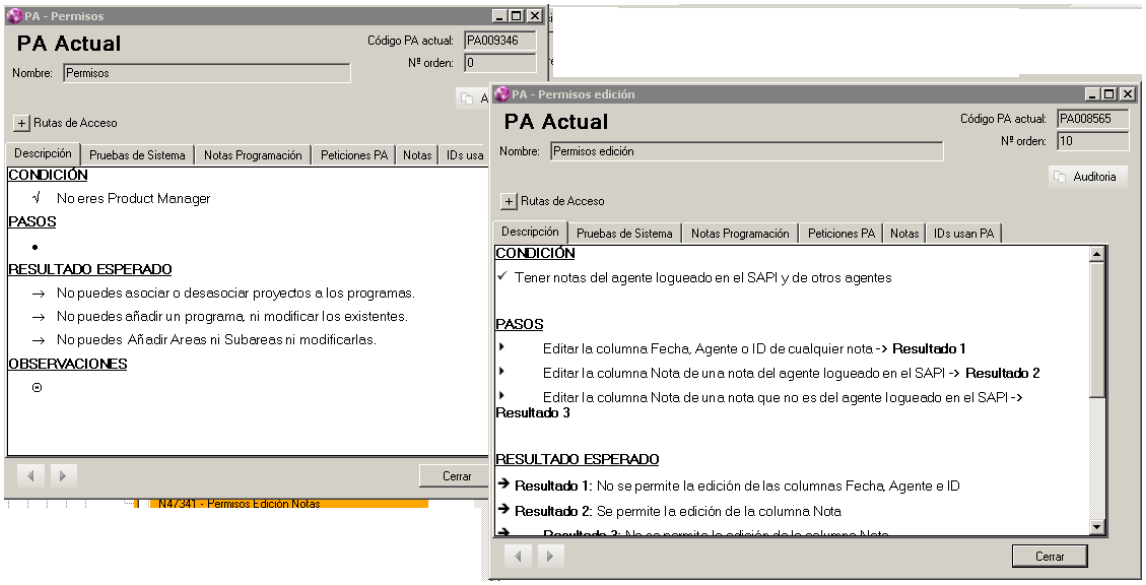


Ilustración 28. Ejemplo de PA que necesita refactorizarse

Es por esto, por lo que necesitamos dentro de nuestro producto, un mecanismo de mantenimiento por el cual nos permita refactorizar[2] aquellas pruebas actuales del sistema que no están bien definidas o añadir pruebas actuales donde su comportamiento sí que exista en el producto pero no esté definido en el gestor de requisitos del mismo.

[2] El termino refactorizar se utiliza en el ámbito del código para realizar mejoras en la estructura del código. En esta tesis extrapolamos el concepto al ámbito de los requisitos expresados como PAs, con lo que la palabra nos sirve para definir la acción de mejorar la especificación de requisitos sin cambiar el comportamiento

Otra motivación importante del desafío es rebajar la carga de trabajo que tienes los analistas y estimar correctamente los tiempos de los analistas dedicados al análisis de WU. Actualmente cuando definen las PAs que compondrán una WU si veían alguna PA que estaba mal definida, la marcaban de modificación y editaban su descripción esto llevaba los problemas de control de tiempo, ya que el tiempo dedicado al análisis, había que sumar el tiempo extra dedicado al mantenimiento de pruebas. Con esta nueva funcionalidad se podrá separar el tiempo del analista dedicado al análisis de WU y el tiempo dedicado al mantenimiento de las pruebas del gestor de requisitos.

3.3.2 Objetivo

El objetivo principal de este desafío es la incorporación dentro de TUNE-UP de un mecanismo sencillo que permita a los analistas añadir comportamiento ya existente en el producto pero que no está definido dentro del gestor de requisitos, además también añadir la posibilidad de cambiar la especificación de una prueba actual sin que esto conlleve cambiar el comportamiento del producto.

Como objetivos secundarios, cuando el trabajo de refactorización o añadido de pruebas actuales faltantes no se hagan en esa misma WU, debe de existir un mecanismo donde se quede marcado claramente que nodo o prueba actual es necesario refactorizar, para que se tenga en cuenta en futuras WUs, esta marca debe de quedar visible hasta que se haya realizado correctamente el proceso de refactorización o de añadido de pruebas faltantes.

Es importante también el registro de una explicación donde se exponga con claridad el motivo que tiene el analista para marcar el nodo o la prueba actual como pendiente de refactorizar[2] o añadir pruebas actuales.

Todo el trabajo de refactorización o añadido de pruebas faltantes debe de ser compartido por todos los analistas por lo que debe de existir de un mecanismo que permita visualizar una lista de aquellas pruebas que han de refactorizarse.

3.3.3 Diseño

Detallados los objetivos anteriormente procedemos en la explicación del diseño para llevar a cabo nuestro objetivo, disponer de un mecanismo de mantenimiento de pruebas actuales dentro TUNE-UP.

Antes de empezar es importante conocer como esta estructurada la herramienta TUNE-UP. La herramienta nos permite trabajar con WUs que serán definidas dentro del gestor de WUs, las WUs están expresadas en pruebas de aceptación que se definen dentro del gestor de requisitos. El gestor de requisitos permite definir las pruebas para una WU, las pruebas las clasificamos dentro de sus respectivos nodos, un nodo representa una parte funcional del producto. En la Ilustración 29 se puede ver el gestor de requisitos donde a la izquierda tenemos el árbol de nodos y en la parte inferior derecha se puede observar el listado de pruebas.

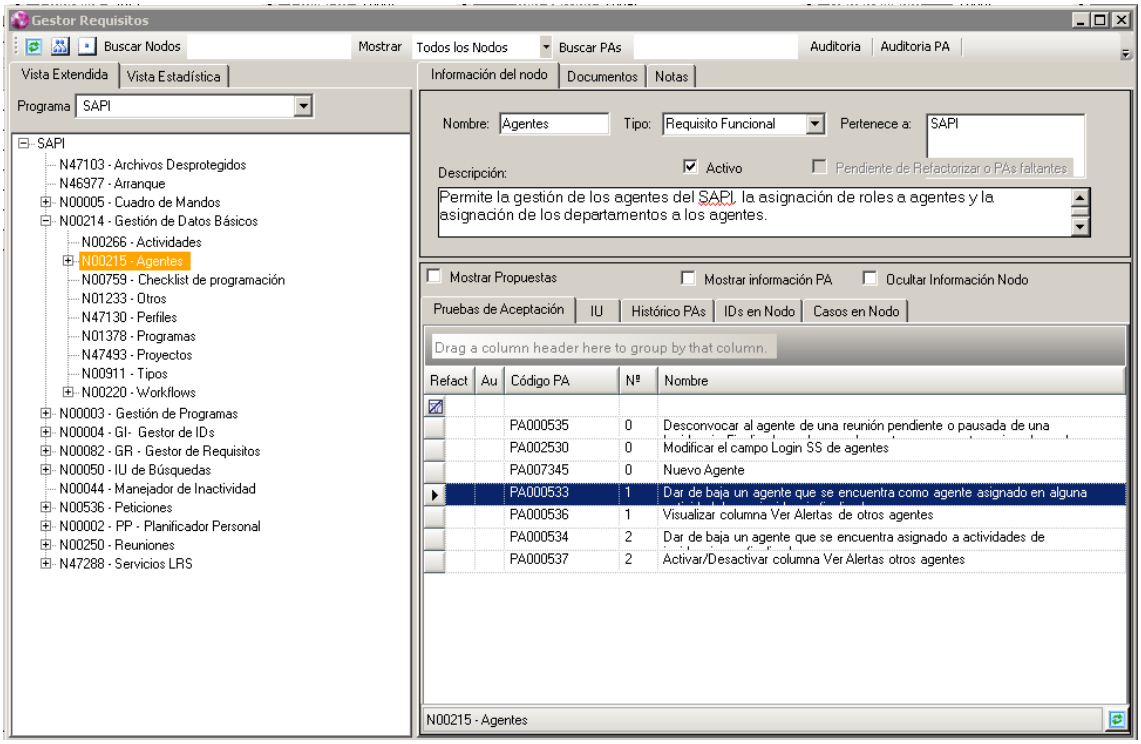


Ilustración 29. TUNE-UP visión de unidades de trabajo

Una vez establecido la estructura y teniendo claro los conceptos de gestor de requisitos y gestor de unidades de trabajo podemos comenzar con el diseño.

El diseño lo hemos dividido en varios bloques dependiendo del nodo que se ha de modificar para aplicar los cambios, además hemos añadido en el diseño algunos cambios que se van a realizar sobre la base de datos.

Uno de los objetivos es el de que lo analistas puedan realizar una marca de aquellos nodos que tengan PAs que necesitan refactorizarse o hay PAs faltantes dentro del nodo, para poder realizar esto, la marca ha de ser guardada en la base de datos, ya que esta marca debe de ser visible mientras no se haya hecho el proceso de refactorización, por ello se debe de añadir en la tabla donde contiene la información de los nodos GRNodo una nueva columna de tipo booleano llamada PendienteRefactorizar cuyo valor por defecto será false.

Con la nueva columna solo se tiene guardada la marca a nivel de nodo pero debemos de hacer lo mismo a nivel de pruebas actuales, por lo tanto también se ha de añadir una nueva columna de tipo booleano para guardar la marca, en este caso se hará en la tabla GRAceptacion.

Otra de las cosas a tener en cuenta en la base de datos es que debemos de poder registrar alguna descripción o comentario asociado a la marca, esto además nos va a servir para poder añadir notas a las pruebas y notas en los nodos, por ello se van a crear dos nuevas tablas en la base de datos llamadas GRNotasPAceptación y GRNotasNodos estas dos nuevas tablas deben de poder guardar información sobre quien ha realizado la nota, cuando la realizo, así como que idNodo o IdPAceptacion pertenece la nota.

Los cambios realizados anteriormente solo afectarían a la base de datos, los cambios que se detallaran a continuación se realizaran sobre la interfaz.

Hasta ahora solo hemos hecho el diseño para poder almacenar la marca y registrar las explicaciones del porqué un nodo o una PA es marcada para refactorizar, pero

necesitamos algún mecanismo para que desde la interfaz podamos realizar la marca.

A nivel de Nodo se va añadir un checkbox dentro del usercontrol que muestra la información del nodo, no solo requiere el checkbox si no un sistema que guarde y actualice el estado del nodo, además debe de respectar los permisos de edición de los demás elementos dentro del usercontrol. El nuevo checkbox que utilizara el analista para marcar el nodo como pendiente de refactorización se debe de comportar como nos los ilustra la Ilustración 30: si el nodo no está marcado y lo marcamos, se abrirá un pop-up que nos permitirá introducir una explicación, al darle a finalizar el pop-up guardara la nota, es decir, guarda el agente que introdujo la nota, la fecha y el nodo, además en el árbol de nodos el nodo quedara marcado con un icono de *warning* que representara que el nodo ha sido marcado para refactorizar o tiene PAs faltantes. Si el nodo ya está marcado y lo desmarcamos, lo primero que hará es comprobar si algunas de sus pruebas actuales estuviera marcada en caso afirmativo deberá de salir un mensaje que impida quitar la marca al nodo, en caso negativo se actualizara el estado del nodo en la base de datos, se quitara el icono en el árbol y además de forma automática se debe de crear una nota que indique que el nodo ha sido refactorizado y además se guarde la fecha y el agente que lo desmarco el check.

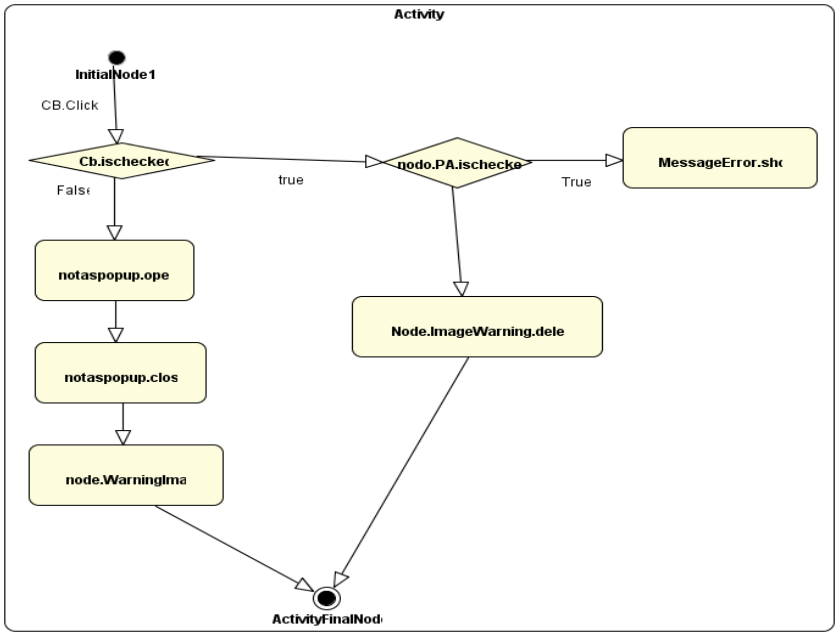


Ilustración 30. Diagrama de Actividad de marcado de nodo

Para poder marcar una prueba como pendiente de refactorizar el proceso será parecido al de marcar un nodo, en vez de añadir un checkbox lo que haremos será añadir una nueva opción en el menú contextual de un prueba actual, la nueva opción nos permitirá marcar o desmarcar la prueba como pendiente de refactorización.

Una vez, una prueba es marcada para refactorizar[2] automáticamente será marcado el nodo o nodos a los que está asociado esa prueba, al mismo tiempo al usuario se le deberá de abrir un pop-up donde será necesario registrar una nota asociada a la prueba, como también se marcará el nodo también se registra en este caso de manera automática una nota para el nodo.

El analista podrá refactorizar[2] la prueba en esa misma ID o en otra, en todo caso una vez refactorizada, se deberá de marcar como refactorizada para ello de deberá seleccionar desde el menú contextual de la prueba la opción, en ese momento se quitara la marca a nivel de prueba pero el nodo seguirá marcado ya que la marca no solo indica que hay que refactorizar[2] las pruebas actuales del nodo sino además que es posible la existencia de pruebas faltantes.

Para poder visualizar si una prueba es necesario refactorizar[2] porque ha sido marcada, se va a incluir una nueva columna de tipo imagen donde a través de la misma imagen utilizada para marcar los nodos veremos que si una prueba ha sido marcada, además si entramos dentro de una prueba que ha sido marcada, esta estará en modo edición y sobre ella aparecerá un label que indicara que está pendiente de refactorización.

Las notas automáticas y no automáticas que se van registrando en la tabla GRNotasNodos pero deberá de existir algún método de visualización de las notas a nivel de nodo, es por esto donde se va a crear una nueva pestaña al mismo nivel que la pestaña de información de nodo, donde se encuentra el check para marcar el nodo, que permita a los analistas observar las anotaciones, no solo anotaciones producidas por el proceso de mantenimiento de PAs si no también otro tipo de anotaciones que se podrán añadir desde la fila de inserción del nuevo grid. El grid de notas ira integrado dentro de un control de usuario ya que se usara el mismo control de usuario para mostrar las notas a nivel de nodo como las notas a nivel de prueba, el grid contara con las siguientes columnas: fecha, agente introductorio, WU a la que va asociada la nota y la nota. El grid además de contar con una fila de inserción para poder añadir más notas contara con una fila de filtrado, que nos permitirá filtrar las notas según sus columnas.

Hasta ahora en el diseño nos hemos centrado en la parte de la refactorización de una prueba, es decir, el cambio de la especificación de una prueba sin que ello conlleve el cambio del comportamiento del sistema, pero esto solo sería el 50% dentro del mantenimiento de pruebas, a veces un nodo contiene funcionalidad dentro del sistema que no está definida dentro del gestor de requisitos del sistema, es por ello que es necesario añadir esa funcionalidad no descrita en el gestor de requisitos, para ello, en el mismo menú contextual del grid de pruebas y propuestas se ha añadido la posibilidad de añadir pruebas faltantes.

No es necesario que para añadir una prueba faltante se marque el nodo, el analista podrá añadirlo directamente, cuando se añade un prueba faltante no es necesario tampoco añadir ninguna nota.

El analista durante el análisis de alguna WU, se dará cuenta muchas veces de la inexistencias de pruebas en el gestor de requisitos que en realidad su comportamiento si que está definido en el sistema, además esta prueba debe de ser vuelta a probar ya que puede verse afectada con el nuevo comportamiento, para poder hacer eso normalmente si la prueba está definida, lo único que tiene que hacer el analista es marcar la prueba como de regresión, en nuestro caso la prueba no existe, por lo tanto se va a añadir en el menú contextual una nueva opción que permita añadir pruebas faltantes de regresión, de esta manera en un único paso se podrá crear pruebas faltantes y además marcarlas para refactorizar[2].

Una vez se ha realizado algún proceso de mantenimiento en el gestor de requisitos sobre algún producto es necesario que ese proceso se revise a través de otro analista, para ello a nivel de programa se ha incluido una nueva pestaña resumen que indica que PAs han sido refactorizada y que PAs han sido añadidas, la nueva pestaña tendrá un grid con información sobre la PA que además de indicarnos a través de un icono que tipo de mantenimiento se ha hecho: refactorización o añadido de prueba faltante, el analista podrá marcar si ha sido revisado la prueba a través de la columna revisado.

Durante toda la parte del diseño se ha comentado que la tarea de mantenimiento de pruebas en el gestor de requisitos es una tarea que se puede hacer a la vez que se analiza una WU, pero a veces debido a la complejidad es necesario posponerlo para ello se ha creado un nuevo workflow desde el cual pueden desarrollar esta tarea o otras los analista, el nuevo workflow llamado WF ticket Analista contendrá tres actividades, introducir WU, realizar ticket analista y revisar ticket analista.

3.3.4 Mantenimiento de PAs en TUNE-UP

Una vez visto el diseño, en esta parte dentro se va a observar como realmente funciona en TUNE-UP el mantenimiento de pruebas a partir de capturas de pantalla de la aplicación.

Para poder realizarlo, vamos a ponernos en la piel de un analista que está analizando una WU desde el gestor de requisitos cuando encuentra una prueba que es necesario de refactorizar[2], lo primero que debe hacer si encuentra una prueba que tiene que ser refactorizada es marcarla como pendiente de refactorización, para ello sobre el menú contextual(Ilustración 31) de la prueba seleccionaremos la opción: Marcar como Pendiente de Refactorizar.

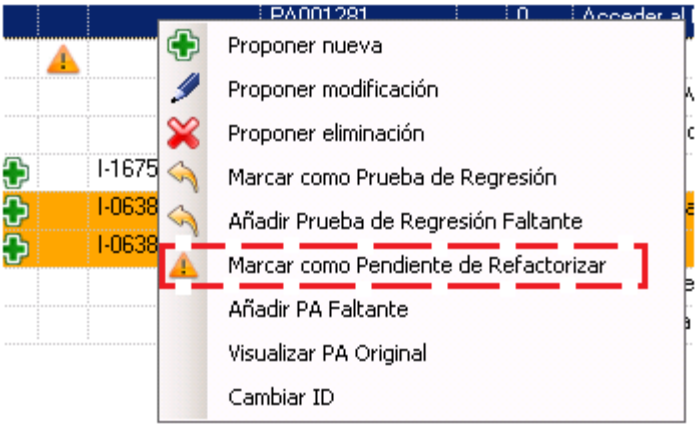


Ilustración 31 Menú Contextual Pruebas de Aceptación

Una vez marcada la prueba el analista debe de introducir una explicación del por qué se ha marcado la prueba como pendiente de refactorización, para ello hará uso del pop-up(Ilustración 32) que se abre inmediatamente después de haber marcado la prueba.

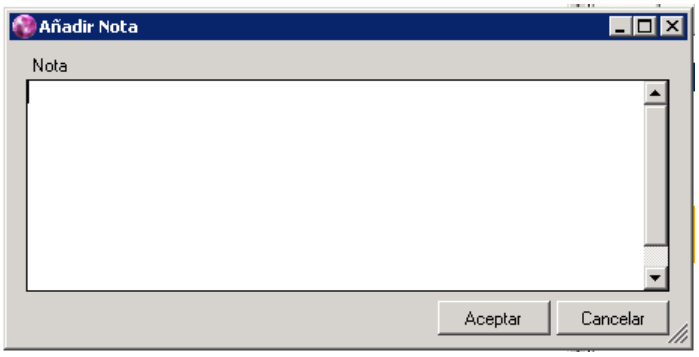


Ilustración 32 : Pop-up de Inserción de nota

Una vez insertada la nota la prueba queda marcada y se puede visualizar(Ilustración 33) gracias al icono de warning que aparecerá en el grid, además el nodo queda marcado con el mismo icono, esto produce también a nivel

de nodo que se introduzca una nota automática con el siguiente texto: "*Nodo pendiente de Refactorizar o PAs faltantes marcado automáticamente al marcar una PA del nodo como pendiente de refactorizar*".

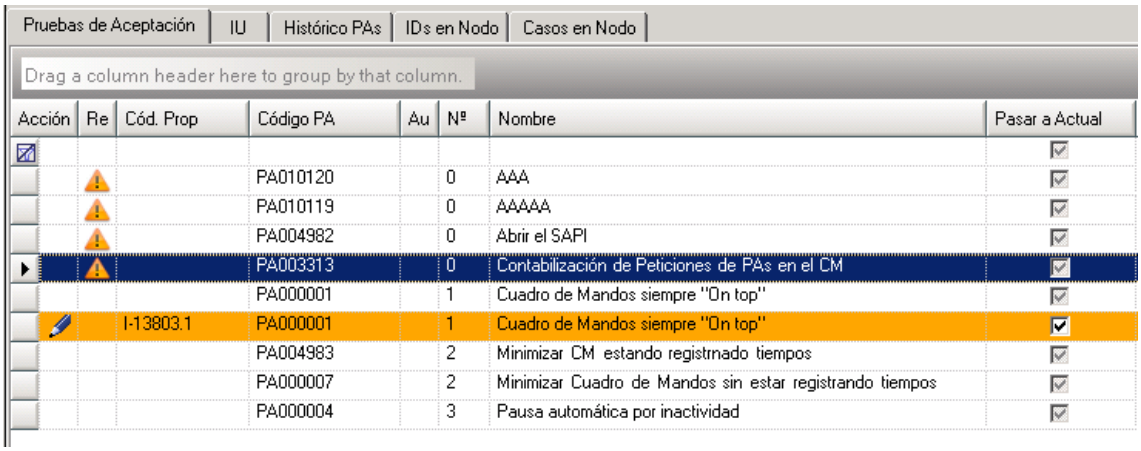


Ilustración 33. Grid de Pruebas de Aceptación con PAs marcadas

También es posible que durante el transcurso del análisis, el analista incluya alguna prueba que no esté definida en el gestor de requisitos pero que su comportamiento esta presente en el producto, para ello desde el menú contextual (Ilustración 34)puede añadir una prueba faltante.

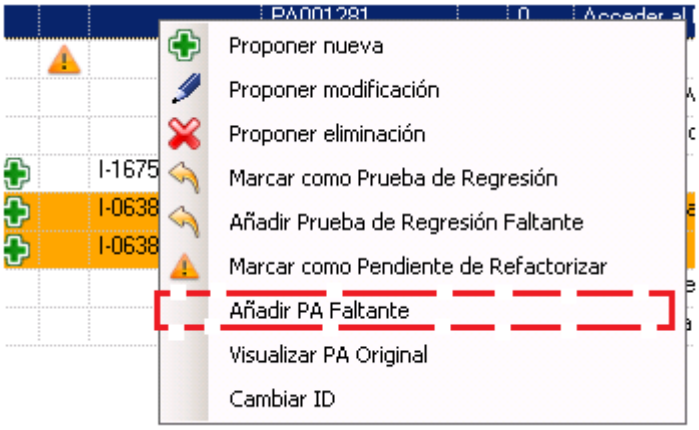


Ilustración 34 Menu contextual PA faltante

Siguiendo con nuestro ejemplo es posible que el analista haya visto alguna deficiencia en algún nodo, bien porque alguna PA tenga que ser refactorizada o bien porque hay alguna prueba que no está definida en el gestor de requisitos pero que existe dicho comportamiento en el sistema, en este caso cuando un nodo

tenga alguna deficiencia comentada anteriormente el analista debe de marcar el nodo como pendiente de refactorizar o PAs faltantes del checkbox en la pestaña de información del nodo (Ilustración 35)

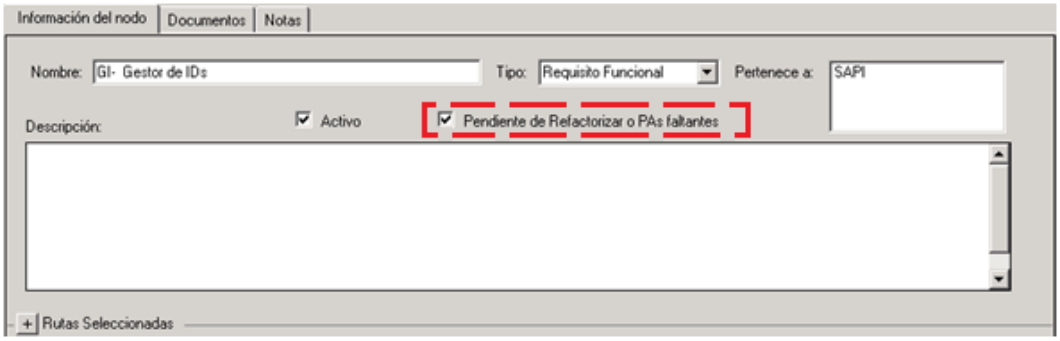


Ilustración 35 Información del Nodo

Al igual que ocurría con una prueba al marcarla como pendiente de refactorizar también es necesario por parte del analista introducir una nota que explique brevemente la decisión de marcar el nodo. Además en el árbol (Ilustración 36) quedará reflejado con el símbolo del warning que el nodo ha sido marcado.

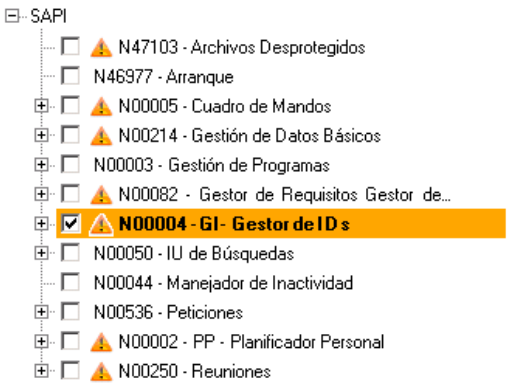


Ilustración 36 Árbol del GR con Nodos marcados

El analista después del marcado puede realizar dos opciones: incluir el mantenimiento en la misma WU que está analizando o posponerlo para más tarde, si decide lo segundo debe de crear un nuevo ticket analista que informará a los analistas. El primero que disponga de la oportunidad cojera el ticket y lo tomara. La Ilustración 37 muestra el workflow asociado a un ticket analista esto permitirá registrar tiempos y por lo tanto separar el tiempo dedicado en el análisis de WUs y el tiempo dedica a la refactorización de pruebas.

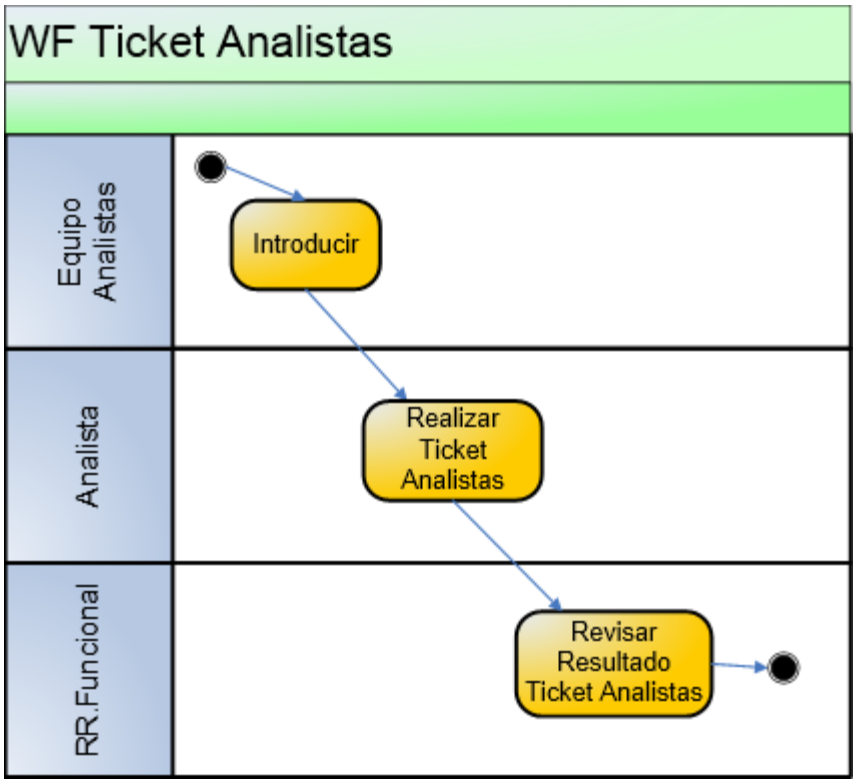


Ilustración 37. WF Ticket Analista

El analista debe de consultar las notas para ver la explicación del marcado pues le ayudará a realizar correctamente el proceso de mantenimiento. Si es el nodo marcado se consultará desde la pestaña notas que se encuentra al mismo nivel que la información del nodo(Ilustración 38). Si el marcado es a nivel de prueba se debe de entrar dentro de la prueba y consultar la nota que se encontrará en la pestaña notas (Ilustración 39).

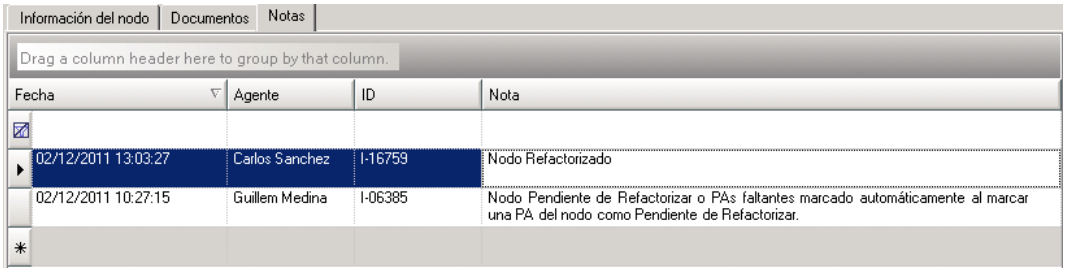


Ilustración 38 Notas a nivel del Nodo

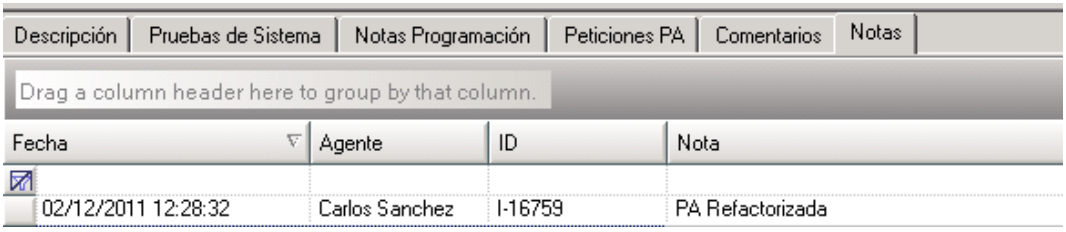


Ilustración 39 Notas al nivel de pruebas.

Una vez realizado el mantenimiento, el proceso se ha de revisar por otro analista, una manera de ver que pruebas han sido añadidas o refactorizadas es ver la lista a través de la pestaña: PAs Refactorizadas o Añadidas en I-XXXX (Ilustración 40). Dicha pestaña se encuentra al entrar en el gestor de requisitos en modo WU y pulsar en el árbol el programa.

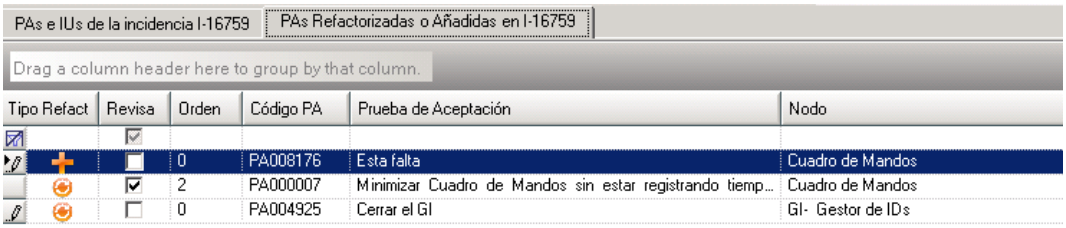


Ilustración 40 Pestaña PAs Refactorizadas y añadidas en I-XXXX

Se puede ver perfectamente qué tipo de refactorización se ha hecho con la columna Tipo Refactorización además de que se pueden ir marcando si ha sido revisada o no la prueba. Una vez marcada como revisada se puede marcar la prueba como refactorizada desde el menu contextual de la prueba(Ilustración 41) o desmarcar el check de refactorización del nodo en la información del nodo.

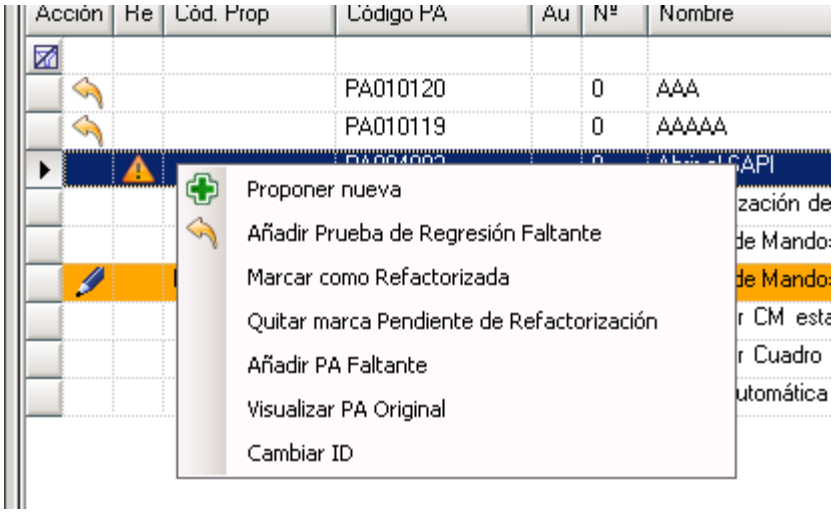


Ilustración 41. Menu contextual Marcar como Refactorizada

Cuando se desmarca un nodo o una prueba queda registrado en forma de nota que analista ha hecho el proceso por lo que siempre queda registrado el trabajo extra del analista.

El motivo por el cual no se desmarcar el nodo automáticamente es porque es posible que no se haya refactorizado completamente ya que puede haber pruebas faltantes.

En resumen, se implementado las siguientes funcionalidades:

- Posibilidad de refactorizar pruebas existentes para mejorar la especificación.
- Posibilidad de añadir pruebas no definidas en el gestor de requisitos pero que su comportamiento está presente en el sistema.
- El analista puede visualizar fácilmente que nodos o pruebas requieren de refactorización.
- El analista puede posponer el mantenimiento de pruebas para futuras WU sin necesidad de realizar el mantenimiento en la misma WU que se detecto el defecto.
- Posibilidad de revisar el trabajo de mantenimiento de manera global desde el propio gestor de requisitos.

- Posibilidad de incluir notas a nivel de nodo y a nivel de pruebas.

En el ANEXO I se podrá ver el documento de PAs que se ha utilizado en el desafío.

Capitulo 4. Coordinación de Proyectos de gran envergadura

4.1. Introducción de Proyectos

En este capítulo vamos a tratar de establecer los conceptos básicos sobre los proyectos desde el punto de vista de la metodología ágil, además veremos la diferencia entre un proyecto tradicional y lo que se conoce como proyecto en la herramienta TUNE-UP.

4.1.1 Proyectos Tradicionales

En este apartado se presentará lo que es un proyecto desde el punto de vista tradicional a partir de sus características.

Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único.[21]

Un proyecto de forma general es un conjunto de actividades que desarrolla una persona o actividad para alcanzar un objetivo. Esta definición anterior se basa en la idea de proyecto desde un punto de vista general, desde el punto de vista tradicional del software podemos decir que un proyecto es la planificación de un conjunto de actividades que se encuentran interrelacionadas y coordinadas, la razón de un proyecto es alcanzar su objetivo dentro de los límites que imponen un presupuesto, calidades establecidas anteriormente y un lapso de tiempo previamente definido.

En el desarrollo de un nuevo producto se inicia estableciendo un objetivo principal que se va dividiendo en un grupo de actividades a realizar para alcanzar el objetivo, además siempre teniendo en cuenta el tiempo para alcanzar la meta y el presupuesto. Una vez una primera versión del producto está realizada y entregada al cliente, se creará nuevamente un nuevo proyecto para añadir nuevas funcionalidades al producto así como mejoras de este, cada nueva entrega está coordinada con la finalización del proyecto que mejora o añade nuevas características al producto.

Un proyecto desde el punto de vista tradicional normalmente tiene duraciones comprendidas entre los 3 meses y dos años, se puede decir que un proyecto se cancela cuando desaparece la necesidad inicial o se agotan los recursos disponibles, es decir, un proyecto puede fracasar si no se tiene una buena gestión del mismo.

La gestión de proyectos es la aplicación de conocimientos, habilidades , herramientas y técnicas a las actividades que engloban el proyecto para poder satisfacer los requisitos del proyecto. La coordinación del proyecto es importante ya que todos los miembros del producto se dedican al proyecto y no solo un grupo de estos.

Un proyecto de estas características siempre está asociado a un único producto y es un emprendimiento de reunir varias ideas para llevarlas a cabo durante un tiempo limitado y que logran un resultado único. Un proyecto finalizara cuando finalizan las actividades que engloban el proyecto, una vez finalizado se produce una entrega al cliente.

4.1.2 Proyectos en TUNE-UP

En el apartado anterior hemos visto el concepto proyecto desde el punto de vista tradicional pero el concepto de proyecto se adapta cuando nos referimos a un proyecto definido en TUNE-UP

Hemos visto que el concepto proyecto desde el punto de vista tradicional se refiere a la idea de elaborar un nuevo servicio dentro de un producto que normalmente coincide con la entrega de una nueva versión del mismo.

Cuando se viene de una metodología tradicional a una metodología ágil es importante conocer los nuevos conceptos ya que son necesario para adaptarse al nuevo enfoque.

En la metodología ágil lo importante es el hecho de hacer entregas al cliente lo más frecuente posible, estas entregas o versiones del producto se conocerán como *release*, el desarrollo de estas *release* esta organizado por iteraciones las cuales llaman sprints y no suelen durar más de un mes. Claro está que normalmente la primera entrega al

cliente rara vez se consigue en una sola iteración debido a que la primera entrega debe de tener al menos un nivel operacional mínimo. A partir de la obtención del producto con un mínimo nivel operacional para el cliente, el producto pasa a producción y es desde ese momento cada sprint podría efectivamente generar una nueva versión factible de ponerse en producción.

Esta primera release hasta que el producto pasa a producción coincide con la idea tradicional del concepto de proyecto, claro está que a partir de ese momento en el que producto pasa a producción no se puede decir proyecto de manera tradicional a las siguientes entregas del producto, sino mas bien llamarlas versiones o simplemente entregas.

Entonces, ¿qué es lo que se conoce como proyecto en una metodología ágil?. Se llama proyecto dentro del contexto iterativo e incremental de una metodología ágil, cuando se realiza un cambio de envergadura en el producto, lo que lleva a la necesidad de organizar el cambio en distintas iteraciones del producto hasta la finalización de los cambios considerados.

En el contexto ágil el término proyecto es cercano al término theme. Un Theme es un conjunto de cambios relacionados que puede afectar a uno o varios productos. A menudo un Theme tiene que ser dividido por varios sub-themes que al final acaban en una parte del producto. Los Themes están compuestos de Epics relacionadas en cuanto a la funcionalidad o características, un Epic no son mas que historias de usuario (User Story) demasiado grandes como para que este únicamente en una sola iteración, a menudo el término Epic es usado cuando una historia de usuario necesita ser partida.

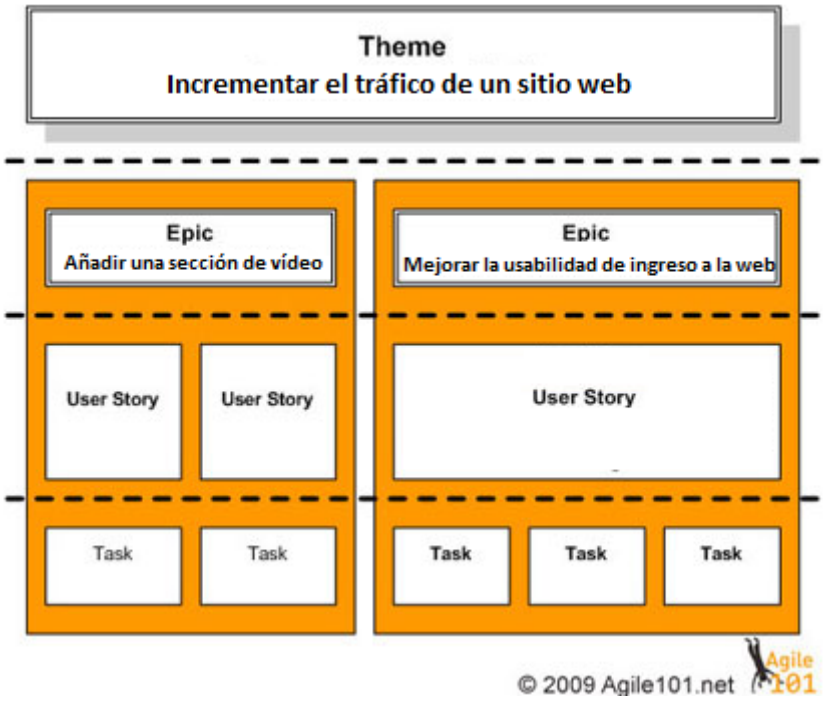


Ilustración 42. Jerarquía entre Theme, Epic and User Story

A diferencia del concepto de proyecto a nivel tradicional, en un Theme (concepto de proyecto en metodología ágil) es posible la intervención de varios grupos de trabajo, o incluso a veces es un trabajo que engloba no solo un producto sino varios productos, esto ocurre cuando desde un producto tiene partes que enlazan a otro producto distinto. En estos casos la coordinación se dificulta ya que cada grupo tiene su propio Product Manager, por lo que es recomendable un nuevo rol, el rol del Project Manager. Mientras que el Product Manager esta más orientado a los clientes del producto y el trabajo persiste mientras existe el producto, en el Project Manager su trabajo se centra en la gestión exitosa del proyecto, la cual debe de estar alineada con la visión del producto o productos, en el caso de Project Manager su duración es la duración del proyecto.

Cuando un proyecto está asociado a un único producto, la coordinación se hace interna al producto y normalmente participa un único grupo de trabajo que además son miembros del producto. Cuando un proyecto tiene varios productos asociados , se requiere de alguna herramienta para facilitar esta coordinación, en el siguiente capítulo presentamos el gestor de proyectos de TUNE-UP y técnicas que nos ayuden alcanzar nuestro objetivo, la gestión coordinada y exitosa dentro de un proyecto, cuando está asociada más de un producto.

4.2. Gestión de proyectos en TUNE-UP

4.2.1 Motivación

En el capítulo anterior vimos las características que tenía el concepto de proyecto desde una metodología ágil. Vimos a modo resumen que un proyecto es un conjunto de Epics que estaban relacionadas entre sí, donde normalmente estas Epics eran de gran envergadura y tenían que ser divididas en varias historias de usuario. Normalmente estos proyectos, Themes a partir de ahora para diferenciarlos de los proyectos tradicionales, normalmente se desarrollan en más de una iteración del producto.

La coordinación de los equipos es un aspecto importante y por lo tanto es necesario el apoyo de alguna herramienta que ayude al Project Manager en la gestión del proyecto, similar al apoyo en el product backlog para el Product Manager.

Además de la coordinación, otra dificultad es saber lo que tiene que hacer cada grupo y para cuándo lo tiene que terminar. Se necesita una herramienta de visualización global de los proyectos donde cada integrante del equipo pueda consultar de una forma rápida qué actividades debe de realizar.

En este nuevo desafío lo que se propone es el desarrollo de una serie de utilidades y protocolos que sean utilizados para la gestión correcta del proyecto dentro de la herramienta TUNE-UP. Estas herramientas ayudarán al Project Manager y a los diferentes grupos de trabajo a gestionar las diferentes WU.

4.2.2 Objetivo

El objetivo de este desafío es la generación de técnicas y utilidades que permitan una gestión adecuada de los proyectos en la herramienta TUNE-UP.

Se necesitan para relacionar proyectos con productos y viceversa. Esto nos permitirá conocer que proyectos van asociados a un producto, así como que productos van asociados a un proyecto.

Una vez establecida dicha relación, el usuario asigna una WU a un producto, por lo tanto una WU no solo estará relacionada con un producto sino que podrá estar relacionada con un proyecto.

Hay que tener en cuenta también que una WU puede estar relacionada con otras, siendo esta contenedora de muchas WUs más pequeñas o que esté contenida en una WU mucho más grande.

Además, un proyecto debe poder visualizarse de una forma práctica al usuario, deben visualizarse las WUs que contiene un proyecto, así como las WUs contenidas y contenedoras de ese proyecto, es decir, visualizar la relación padre-hijo entre WUs que intervienen en el proyecto.

Necesitaremos de una funcionalidad en la herramienta que nos muestre dentro de un proyecto qué WUs son las que realmente tienen más peso para que el Project Manager pueda repartir los recursos adecuadamente entre WUs del proyecto.

Por otra parte, debemos contar con una técnica que nos facilite el control del proyecto.

Todas estas necesidades deben ser cubiertas con funcionalidad dentro de la herramienta TUNE-UP y en un contexto ágil.

4.2.3 Diseño del Gestor de Proyectos

La Ilustración 43 podemos visualizar el diagrama de clases para ver las relaciones que debemos construir en la base de datos. Una WU estará asociada con un producto. Un producto contendrá varias WUs. Una WU está compuesta por varias PAs. Lo mas importante será establecer la relación muchos a muchos entre proyectos y productos,

un producto podrá tener varios proyectos y un proyecto podrá implicar a varios productos.

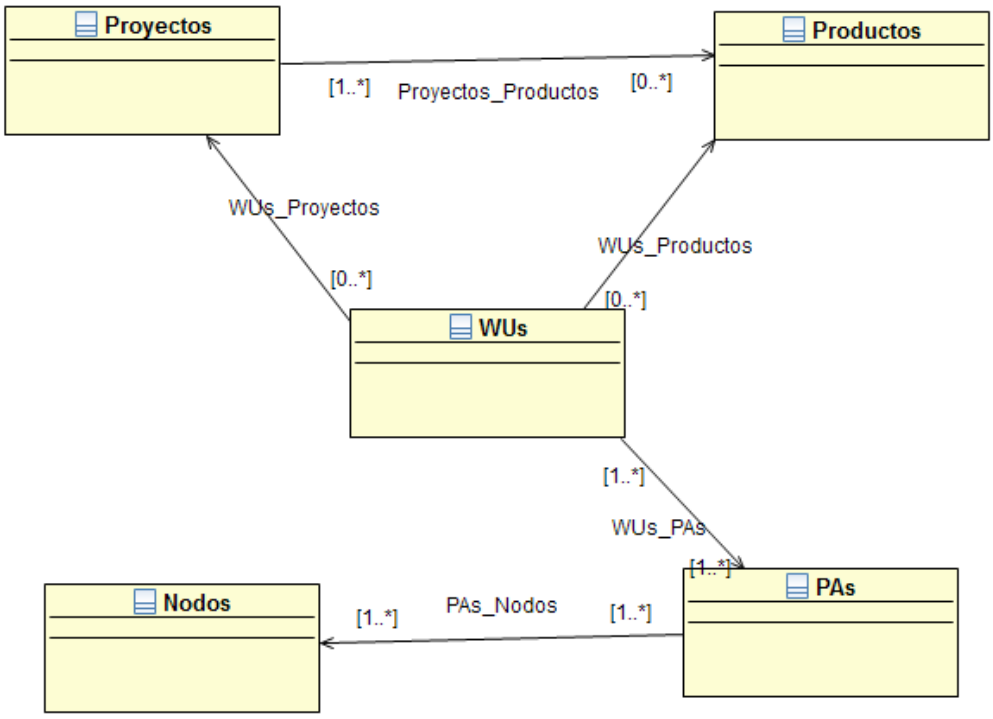


Ilustración 43 Diagrama de clases WUs, Proyectos, Productos

Para poder realizar esta asociación dentro de la herramienta TUNE-UP, se va a crear una nueva pestaña dentro de gestión de datos básicos llamada proyectos, desde esta pestaña se visualizará un grid donde podremos ver el nombre de los proyectos, si están activos y una descripción de los mismos, podremos también asignar agentes a los distintos roles de un proyecto así como establecer relaciones de los productos a proyectos. Un proyecto se podrá borrar si no tiene ninguna WU asociada. Se podrán añadir nuevos proyectos siempre que el proyecto no esté ya existente.

Al igual que la pestaña proyectos, en la pestaña productos se ha añadido también la posibilidad de asociar un proyecto al producto.

Tanto la asociación de un proyecto a producto que aparece en la pestaña proyectos como la asociación de un producto a proyecto que aparece en la pestaña producto es la misma, lo único que cambia es la manera de mostrarse, mientras que uno se

muestran la lista de productos asociados a un proyecto, el otro muestra una lista de proyectos asociados al producto.

Hay algunas restricciones a la hora de asociar y desasociar un proyecto de un programa, por ejemplo no se podrá desasociar un proyecto si existen WUs asociadas a ese proyecto y a ese producto.

En el planificador personal de la aplicación TUNE-UP se han mejorado los filtros de acuerdo a las asociaciones, es decir, no se va a puede aplicar un filtro que contenga un producto y un proyecto si no existe esa relación.

Lo mismo va a ocurrir en otros filtros que se apliquen como en el formulario de búsquedas de WUs.

Cuando se introduce una nueva WU es necesario establecer el producto al que va asociado la WU, es opcional introducir el proyecto, porque puede ser que la WU sea de carácter general al producto y no algo concreto de un proyecto. En el caso de seleccionar un proyecto solo se mostraran aquellos proyectos que estén asociados con el producto de la WU.

En el gestor de WUs para contemplar las relaciones que una WU puede tener con otras WUs se hará uso del grid de la pestaña relaciones, donde se establecerán las relaciones que una WU puede tener con otras. Las relaciones que más nos interesan son las de Contenedora y contenidas.

Una WU contenedora es una WU que es demasiado grande y se ha fragmentado en otras WUs más pequeñas, esta WU contenedora puede ser que también este contenida en una WU mucho mayor.

Uno de los objetivos más importantes es el de ver de forma rápida y lo más clara posible que WUs están asociadas a un proyecto, para ello se va a crear el formulario de proyectos.

El formulario de proyectos está compuesto por un combo, donde poder seleccionar el proyecto, el combo contendrá todos los proyectos que están activos. Una vez el usuario seleccione uno de los proyectos se visualizara todas las WUs que están

asociadas a ese proyecto. Las WUs se mostrarán en un árbol donde se podrán ver las relaciones de composición entre las WUs del proyecto seleccionado, para poder realizar esta operación se realiza en dos pasos: el primer paso es establecer una consulta donde se devuelvan solo las WUs que son contenedoras y a continuación otra con las WUs contenidas, a continuación se hace un mecanismo de relación entre los valores devueltos por la consulta de manera que nos crea un árbol donde se podrán ver las relaciones padre-hijo de las WUs.

El árbol se muestra en forma de grid para poder ver la información de la WU en forma de tabla donde sus columnas representan la información de la WU. Las columnas más importantes serán las de ID, producto, versión, descripción, orden, Agente Introductorio de la WU y la Actividad Actual del workflow en la que se encuentra la WU.

Cada fila del grid será seleccionable y para poder consultar la información concreta de la WU, podremos ir al gestor de WUs haciendo doble clic sobre la columna ID.

Algunas columnas del grid podrán ser modificables como el orden y la descripción, esto permitirá al Project Manager cambiar algunos de los valores de la WU sin necesidad de acceder a ella a través del gestor.

Al ser el grid seleccionable, se podrá seleccionar más de una fila y por lo tanto se podrá ir al gestor de WUs a partir de una lista de WUs seleccionadas.

El formulario de proyectos también nos va a dar una medida del número total de WUs que tiene un proyecto, para ello se utilizarán dos contadores que serán recalculados cuando se cambia de proyecto en el combo. Se utilizan dos contadores uno para saber el número de WUs totales del proyecto y otro para contar el número de WUs hijas que contiene una WU seleccionada.

A veces el número de WUs dentro de un proyecto puede ser bastante elevado por lo que necesitamos incorporar una serie de filtros que nos permitan filtrar las WUs.

Uno de los filtros que interesarían al Project Manager serian el filtrado por versión, una WU esta asignada a una versión del producto por lo que es interesante saber exactamente en que versión del producto debe de publicarse la WU. El filtrado por

versión también incluye el filtrado por producto, por lo tanto a la hora de filtrar por versión es necesario también decir para que producto.

Otro filtro que incluirá es el filtrado por agente, por cuestiones importantes y de consulta es importante saber quién introdujo la WU, por si está no estuviera completa o su descripción no fuera coherente, esto nos permitirá poder establecer una reunión con el agente introductorio que permita el aclarado de la WU.

Otro filtro que no podía faltar es el filtrado por contenidos en la descripción, este filtro permite saber si una WU esta repetida o tiene algún tipo de relación de seguimiento.

El panel de filtrado debe de estar oculto y se mostrará cuando se pulse sobre el botón mostrar filtros. Se ocultará cuando se pulse el botón ocultar filtros. El hecho de que sea un panel oculto es por qué la utilización de los filtros será en un momento puntual y podemos quitar parte de la visualización del árbol de las WUs de un proyecto.

El panel de filtrado además de tener los campos de filtrados debe de disponer de dos botones, uno para filtrar y otro para borrar los filtros establecidos. Además en cada uno de los campos de filtrado comentados anteriormente, se dispondrá de un tooltip que permita al usuario de manera rápida ver realmente los criterios de filtrado.

Puede ser que uno de los filtros se cumpla en una WU hija pero no en el padre para ello al filtrar, se colorearán en un tonto pastel las hijas que cumplan el filtro dejando el padre en un color blanco. No se visualizan hijas que no cumplan el filtro, solo se visualizan padres que tienen hijos que cumplen el filtro.

En la sección siguiente veremos algunas capturas de pantalla que aclararan lo que se está comentado en esta sección.

El formulario de proyectos contendrá un botón llamado visualización que nos permitirá visualizar las WUs contenidas en el proyecto en un TreeMap, esta técnica de visualización junto con otro tipo de técnica se detallaran más adelante.

Como habéis podido leer en este capítulo se ha explicado el diseño de un conjunto de utilidades que implementaremos en TUNE-UP para la gestión correcta de los

proyectos, en el capítulo siguiente se detalla a partir de capturas de pantalla el funcionamiento del diseño establecido en esta sección.

4.2.4 Gestor de Proyectos en TUNE-UP

En este capítulo se va a presentar la gestión de proyectos dentro de la herramienta TUNE-UP, se explicará paso a paso desde como introducir un nuevo proyecto hasta la gestión del mismo desde el formulario de proyectos en la pantalla principal.

Los proyectos surgen de la necesidad de realizar un conjunto de WUs relacionadas y cuyo tiempo excede en más de una iteración del producto o cuando WUs de diferentes productps deben sincronizar su implantación. Inicialmente se parte de un conjunto de WUs que van siendo fragmentadas en WUs más pequeñas, normalmente lo que se hace es crear una WU contenedora que agrupa las WUs más pequeñas.

Una WU puede estar relacionada con otra WU de distintas formas, se dice por tanto que hay distintos tipos de relaciones. Una WU puede estar relacionada por composición, es decir, que una WU este dentro de otra. Otro tipo de relación es de continuidad es decir que una WU continua en otra o relaciones de importancia que indiquen que realizar una WU es importante para esta otra WU.

En el formulario de proyectos de momento no necesitamos las relaciones horizontales, el único tipo de relaciones que nos interesan son las relaciones de composición para poder ver la relación de padre e hijos entre WUs

En la Ilustración 44 tenemos la vista clásica del gestor de WUs, en la ficha de la WU estableceremos los parámetros y una descripción, esta información será muy importante para los agentes que realicen cada una de las actividades del workflow dentro de la WU.

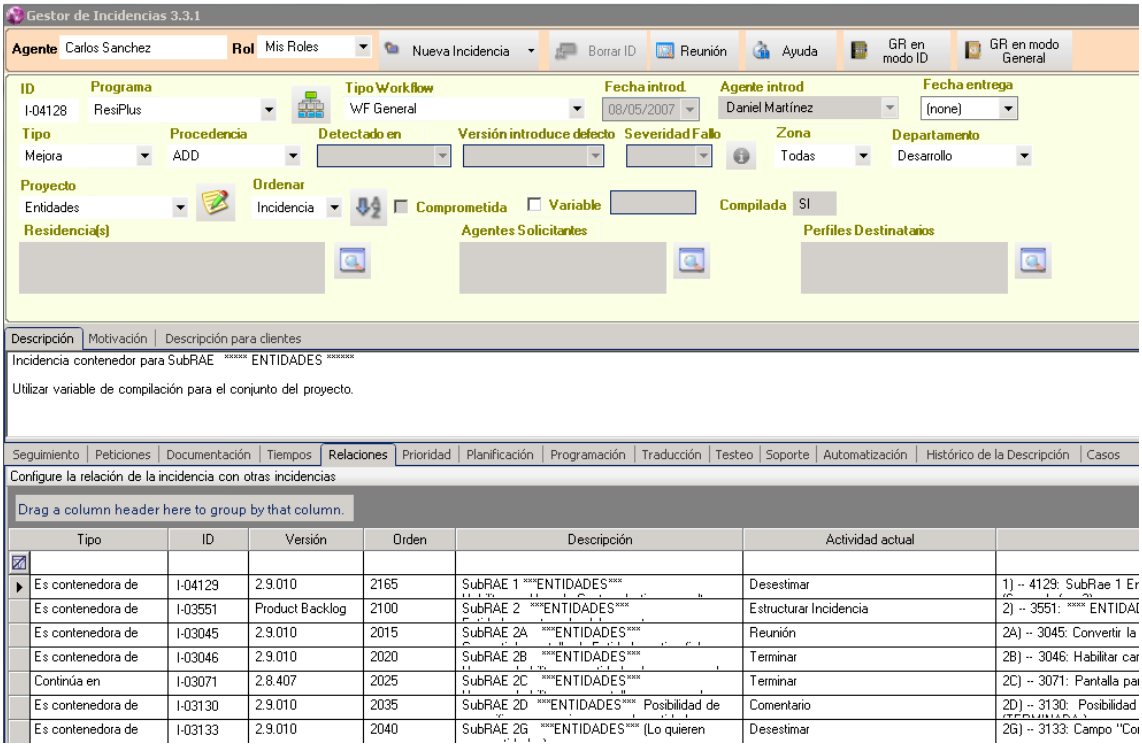


Ilustración 44. Gestor de WUs TUNE-UP con Pestaña de Relaciones Abierta

Para poder establecer las relaciones utilizamos el gestor de WUs la pestaña relaciones que podemos visualizar en la Ilustración 45. La pestaña relaciones muestra un grid con las diferentes relaciones que tiene una WU con otras WUs, para establecer una relación la única información que se necesita es indicar el tipo de relación y el código de la WU con la que se relaciona, una vez puesto esos campos las demás columnas del grid se rellenan de manera automática.

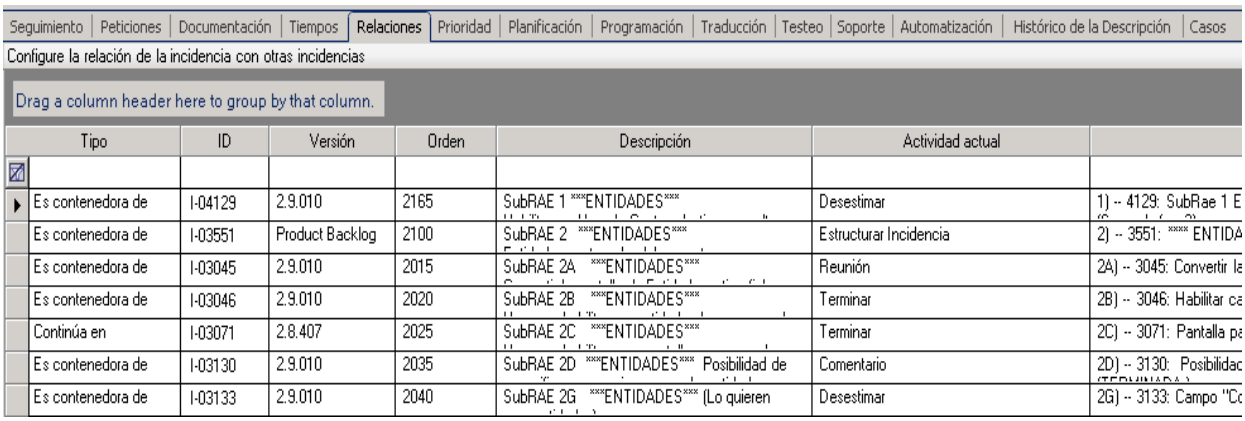


Ilustración 45. Tipos de Relaciones de una WU

Cuando el Product Manager prevé que un conjunto de WUs que están relacionadas entre sí es demasiado grande para realizarla en una simple iteración debe de crear un proyecto.

Para crear un nuevo proyecto el Product Manager debe de realizarlo desde el formulario de gestión de datos básicos, la pestaña de proyectos. Para registrar un nuevo proyecto es necesario introducir un nombre y un estado. El nombre no debe de estar repetido y opcionalmente se podrá incluir una descripción del proyecto. La Ilustración 46 muestra el gestor de datos básicos la pestaña proyectos.

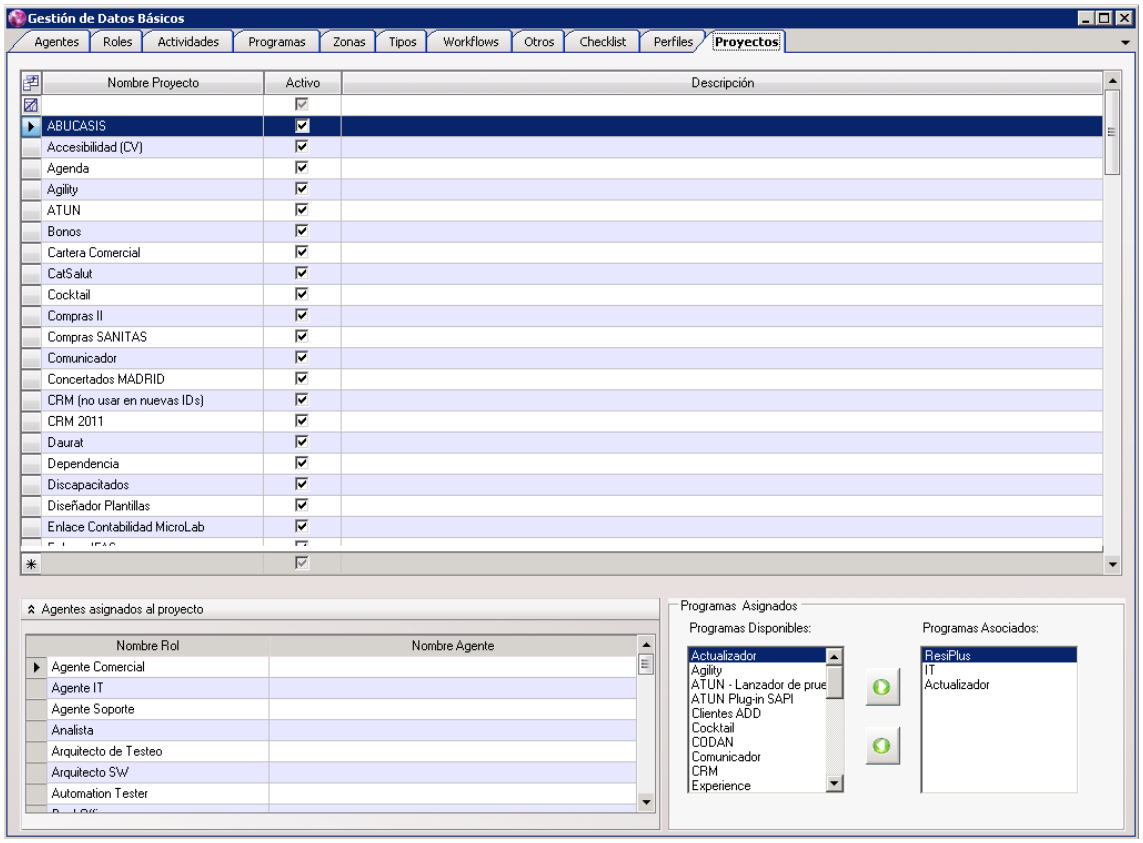


Ilustración 46 Gestión de datos básicos pestaña proyectos.

Una vez introducido el proyecto, es necesario indicar con qué productos va a estar asociado el proyecto, esto se hará desde la misma pestaña en la parte inferior izquierda donde encontrara dos listas, una de ellas contiene los productos disponibles y la otra los productos asociados. La Ilustración 47muestra el mecanismo para añadir productos al proyecto seleccionado previamente en el grid.

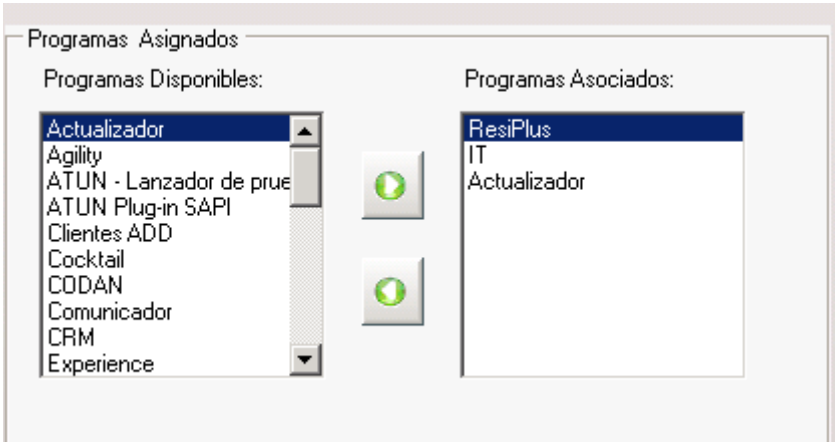


Ilustración 47. Programas asignados al Proyecto

Hay ciertas restricciones llegados a este punto que se deben tener en cuenta a la hora de eliminar un proyecto, un proyecto no se podrá borrar si alguna WU está relacionada con el proyecto, lo que sí se puede hacer es desactivar el proyecto una vez terminado para que no pueda ser relacionado con WUs que no tengan ningún tipo de relación con el proyecto. Lo mismo ocurre a la hora de desasociar un producto del proyecto no se podrá si una WU de ese producto contiene el proyecto relacionado, en ambos casos cuando se produzcan estas restricciones se le informara al usuario debidamente a través de mensajes de información (Ilustración 48).

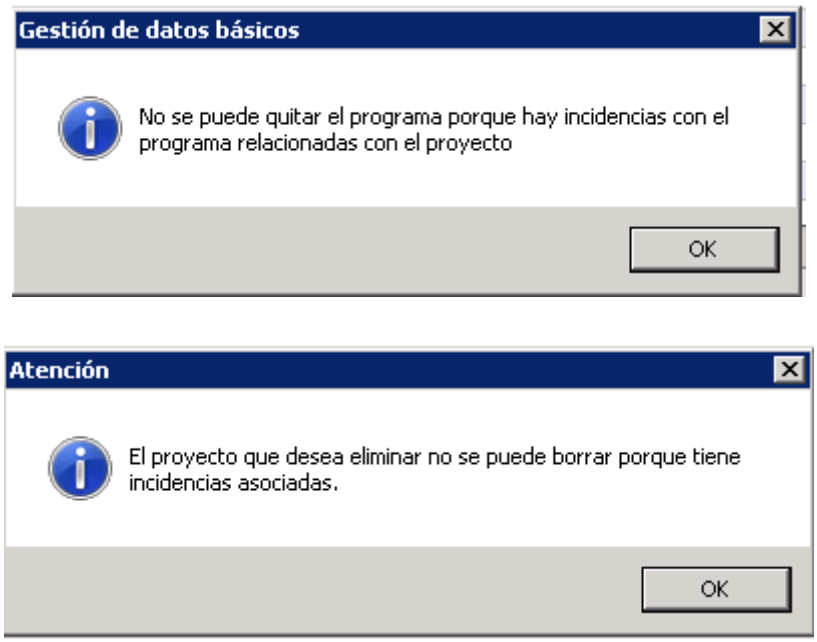


Ilustración 48. Mensajes de Restricción de Proyectos

Llegado a este punto el Product Manager debe de delegar las tareas a un Project Manager que será el encargado de gestionar el proyecto durante la duración del mismo, además debe de asignar al personal que trabajara dentro del proyecto, si es más de un producto varias personas podrán ocupar el mismo rol y se debe de establecer una organización previa con los Product Manager de los otros productos. Para establecer los agentes que participaran en proyecto se asignaran los roles desde la mima pestaña en la parte inferior izquierda(Ilustración 49).

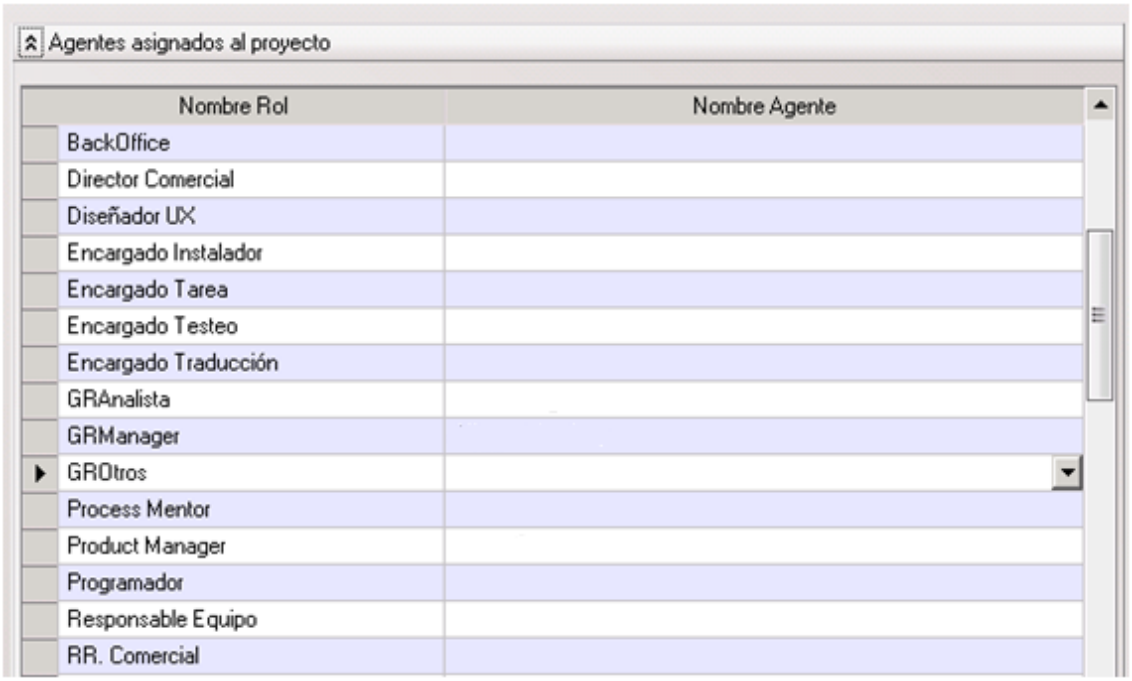


Ilustración 49 Asignación de Agentes al Proyecto

Una vez realizada la asignación del personal se deben de marcar las WUs como pertenecientes al proyecto, esto se hace desde el gestor de WUs la ficha de la WU. Las nuevas WUs que vayan surgiendo del proyecto también deberán relacionarse con el proyecto a través de su ficha(Ilustración 50).

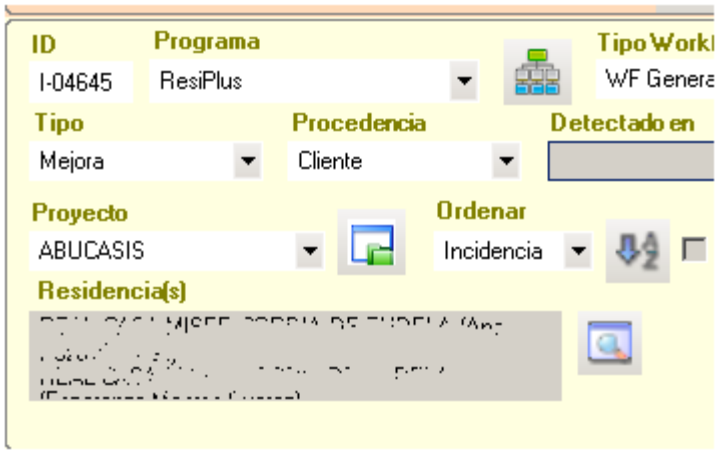


Ilustración 50 Asignación de Proyecto en la Ficha de la WU

Una vez realizado lo anterior, solo quedaría organizarse las estregas de las distintas WUs entre los miembros del proyecto, para esa organización se explicará una técnica más adelante.

Los miembros del proyecto y sobre todo el Project Manager podrá, gestionar el proyecto desde el formulario de proyectos del planificador personal. En este formulario se visualizaran todas la WUs que incluye un proyecto además de las relaciones de contención entre ellas, aunque aparentemente parece un grid en realidad es un árbol que puede ir expandiéndose para poder ver las hijas que esta contiene.

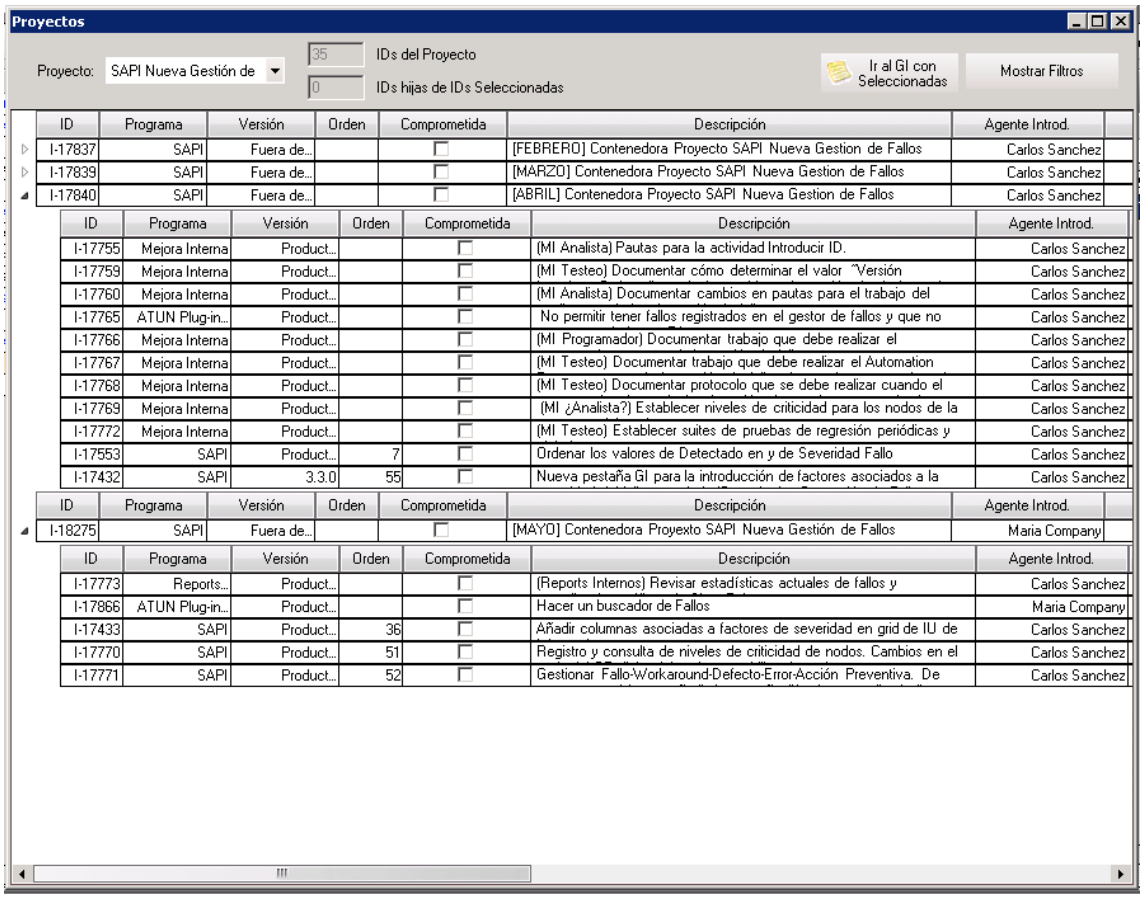


Ilustración 51 Formulario de Proyectos

El formulario de proyectos contiene información útil respecto a las WUs, como su orden, la versión del producto donde se finalizara la WU, la descripción de la misma, la actividad actual en la que se muestra la WU, así como algunas columnas que nos indican el tiempo estimado y el tiempo real invertido en ella.

Desde el formulario de proyectos se puede saber exactamente el numero de WUs que tiene un proyecto, así como el numero de hijas que tiene una WUs, de esta manera se puede contabilizar de forma automática el proyecto en número de WUs que tiene el mismo(Ilustración 52).

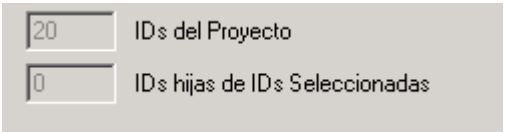


Ilustración 52 Contador de WUs en Proyectos

Desde el formulario de proyectos no solo se puede visualizar un proyecto sino que además se puede realizar un filtrado para mostrar solo aquellas WUs que se desean. Para visualizar los filtros se debe de pulsar el botón mostrar filtros y entonces estos aparecerán en el formulario a la derecha. Se puede filtrar por diferentes criterios: Versión, Agente introductorio y descripción. En un futuro se quiere implementar una forma de filtrado por código de WUs de manera que solo se muestren las WUs que interesen al usuario.

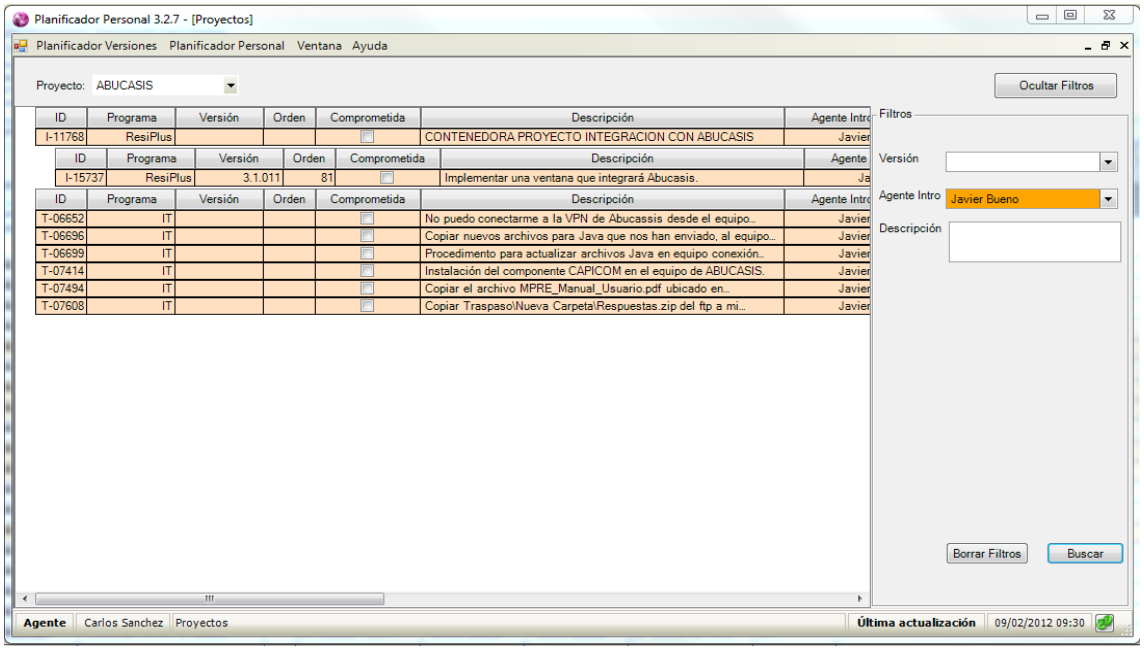


Ilustración 53 Filtrado del Grid de Proyectos.

El grid de formulario de proyectos es realmente un árbol como hemos comentado anteriormente pero además este árbol permite la selección de cada uno de sus nodos, además de la posibilidad de modificar sobre el grid algunos de los campos del mismo como el campo orden utilizado para establecer un criterio de prioridad en las WUs y el campo descripción donde se encontrara la descripción de la WU.

El hecho de que el grid sea seleccionable permite al usuario seleccionar una WU o varias WUs para poder ir al gestor de WUs de las seleccionadas para poder modificar otros campos no disponibles desde el grid.

Se puede visualizar en forma de TreeMap un proyecto a partir del botón Visualizar, en capítulos siguientes se detallará esta técnica de visualizado que permitirá de forma grafica y sencilla de entender que WUs son más pesadas en su desarrollo y por lo tanto más cambios importantes conllevan.

En el siguiente capítulo se va a detallar una técnica utilizada en la gestión de proyectos desde la metodología ágil, el capítulo se divide en dos partes la primera parte explica la técnica y en la segunda parte se detalla cómo fue realmente aplicada dentro de TUNE-UP la técnica.

4.2.5 Creación de Proyectos en TUNE-UP

4.2.5.1 Story Mapping

A la hora de iniciar un nuevo proyecto es necesario armar un buen Product Backlog que permita realizar una planificación efectiva del desarrollo del nuevo proyecto.

El problema es lograr un buen nivel de detalle que permita expresar adecuadamente aquellas tareas que serán más importantes y por lo tantos tengan mayor prioridad.

La técnica de Story mapping[15], desarrollada por Jeff Patton, permite un enfoque visual a la construcción del Product Backlog.

El Product Backlog es una lista donde los elementos aparecen ordenados de mayor a menor valor, mirando más adentro del Product Backlog nos damos cuenta también que en la mayoría de casos los elementos menos prioritarios son los que están menos refinados.

El proyecto será nuestro objetivo, un objetivo es un problema a resolver u objetivo de negocio que se quiere alcanzar, este objetivo será dividido en tareas, las tareas son actividades o tareas que lleva a cabo un usuario para lograr su objetivo, para poder realizar esas tareas necesitaremos de herramientas para soportar las tareas de los usuarios.

Tenemos el objetivo de negocio que será utilizada por una comunidad de usuarios, este objetivo se dividirá en actividades o tareas de usuario y por lo tanto en **user stories** equivalentes al concepto WUs.

Los user stories son polivalentes, los user stories sirven para definir necesidades del usuario, sirven también para definir la descripción del producto o la planificación de un item del producto.

Los user story deben de ser cortos y deben de tener una descripción concisa normalmente siguen la siguiente plantilla: Como usuario [tipo de usuario] , Quiero realizar la siguiente[Descripción de la tarea] para poder conseguir el [Descripción del objetivo], además se pueden incorporar algunas notas, especificaciones o algunos sketches.

La colección de user story representa el product backlog del objetivo de negocio que se quiere alcanzar. El product backlog será una lista de user story priorizada por orden de importancia de los elementos.

Los user story normalmente se descomponen en tareas más pequeñas, estas tareas tendrán como máximo unas 16 horas para terminarlas.

La manera de organizar esos user story son a través de los story maps este tipo de organización nos permiten organizar y priorizar los user story. Las ventajas de un story map frente a un product backlog son:

- Los story maps representan un workflow.
- En los story maps se pueden ver las relaciones entre los user story padres y los user story hijos.
- Los story maps son una herramienta de discusión que nos permite establecer unas pautas para discutir la posición de los user story dentro del story map.

Ilustración 54. Ejemplo de Story Map

Los story maps son modelos multi-dimensionales que nos permiten modelar la secuencia de las actividades y tareas de los usuarios desde el punto de vista de estos. El orden no es necesario que coincida con un orden cronológico, el orden es definido por el proceso de negocio que hay detrás de nuestro sistema. Los story maps también modelan el grado de necesidad de cada mejora que se haga al sistema por lo que aquellas características imprescindibles se encontrarán arriba del todo mientras que más abajo se encontrarán las menos imprescindibles o las que sirven para refinar el producto final.

La construcción del story map es un proceso iterativo y que se debe de realizar con los integrantes del proyecto. Lo primero es establecer el orden de las actividades, las actividades son las cosas que hace el usuario para cumplir el objetivo. Una vez establecido las actividades esas actividades conllevan unas tareas que se han de realizar para llevarlas a cabo. Una actividad puede estar compuestas de varias tareas pero debemos de ordenar las tareas de manera vertical según la necesidad de las tareas.

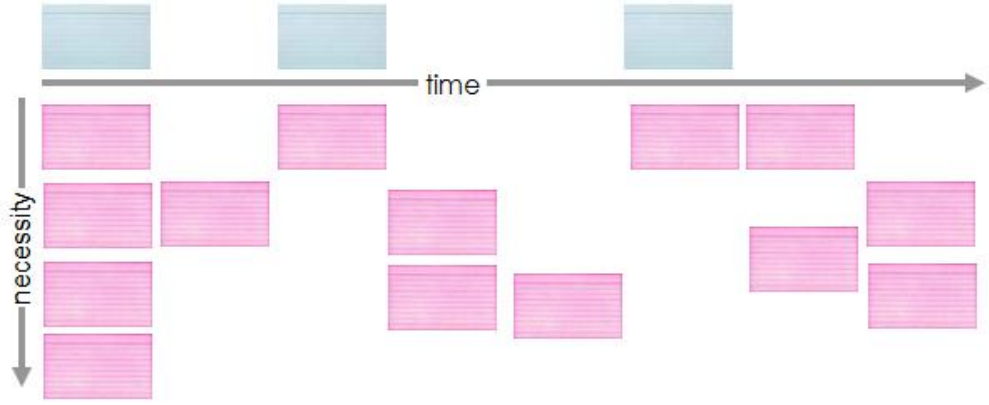


Ilustración 55. Ordenación de las Tareas en sus ejes verticales y horizontales

Una vez establecido las tareas se hace un proceso iterativo para establecer la correcta ordenación de las tareas según sus necesidades. Para poder realizar un buen story maps se necesita de un espacio ya que a veces el número de tareas puede ser muy elevado.

Ilustración 56. Representación "The Walking Skeleton"

Las tareas se pueden dividir horizontalmente para establecer una serie de fases a la hora de cumplir los objetivos. la primera división que se establece es la llamada "The walking skeleton" es la mínima división horizontal que engloban las tareas mínimas para poder sacar la primera versión producto. Una vez establecida esta división se pueden establecer otras divisiones con las otras tareas que se quedan fuera del "The walking skeleton" para establecer nuevas releases o versiones.

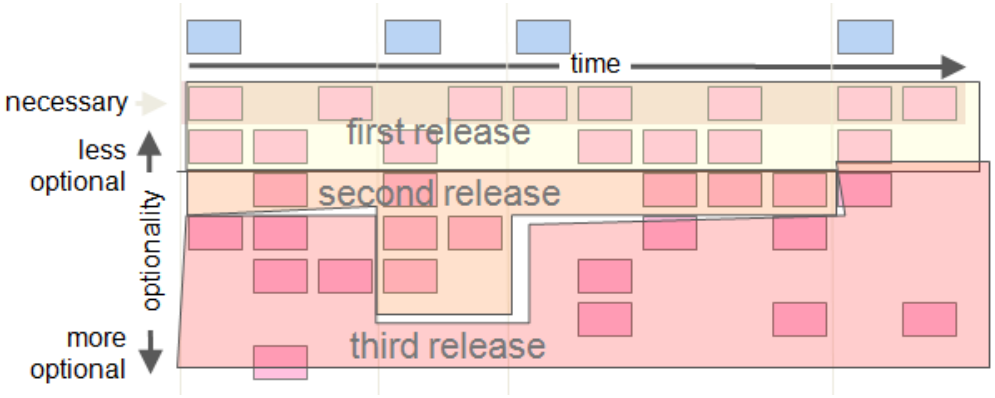


Ilustración 57. División del story map en releases

4.2.5.2 *Story Mapping en TUNE-UP*

A partir de los conocimientos adquiridos con los story maps se plantea la aplicación de los conocimientos en un nuevo proyecto llamado Nueva Gestión de Fallos, como hemos comentado anteriormente un proyecto en TUNE-UP no es únicamente sobre un único producto sino que es aplicado a varios productos, que han de coordinarse para alcanzar un objetivo.

Lo primero que hicimos fue crear una batería de WUs (Ilustración 58) que representarían las tareas dentro del story map, para diferenciar aquellas tareas que pertenecen a un producto o otro estableceremos un código de colores en los post-it.

Ilustración 58. Listado de WUs que comprenden el proyecto "Nueva Gestión de Fallos"

El uso de Post-it en vez de usar tarjetas es por la comodidad de ir pegando y despegando para establecer prioridades de necesidades.

A diferencia del story maps que tiene varias actividades ordenadas horizontalmente respecto al tiempo, nosotros hemos cambiado esas actividades por perfiles destinatarios.

Ilustración 59. Propuesta de Story Mapping del proyecto "Nueva Gestión de Fallos"

Después de un proceso iterativo para ordenar correctamente las tareas según las necesidades pudimos ver claramente aquellas releases que necesitaríamos hasta la terminación del proyecto.

No pudimos establecer realmente las versiones del proyecto ya que al estar involucrados varios productos no sabíamos concretamente que código de versión asignar, lo que sí que hicimos fue establecer unas fechas donde todos los productos y por lo tanto todos los grupos debían de tener hechas las tareas que englobaran ese periodo.

El story map fue evaluado por los distintos componentes de los productos que se veían comprometidos en el proyecto y resultó bastante beneficioso, ya que de un simple vistazo cada grupo sabía lo que tenía que hacer exactamente y para cuándo.

Llegamos a un punto en el que story map lo necesitamos pero no lo podíamos tener en papel sino que necesitábamos el uso de alguna herramienta que guardara la información y que pudiera ser consultada por los miembros del proyecto, por lo que en un primer momento se evaluó la necesidad de construir una nueva herramienta capaz de guardar esta información pero nos dimos cuenta que existía una manera de construir nuestro story map haciendo uso del gestor de proyectos que incorpora la herramienta TUNE-UP.

Como cada actividad estaba ya añadida a la herramienta como una WU, vinculamos cada WU al proyecto, además añadimos el perfil destinatario, una vez hecho esto creamos tres WUs contenedoras que representarían las releases del proyecto (Febrero, Marzo y Abril) y añadimos sus correspondientes hijas, de esta manera desde el gestor de proyectos de la herramienta se puede tener una visión global del story map.

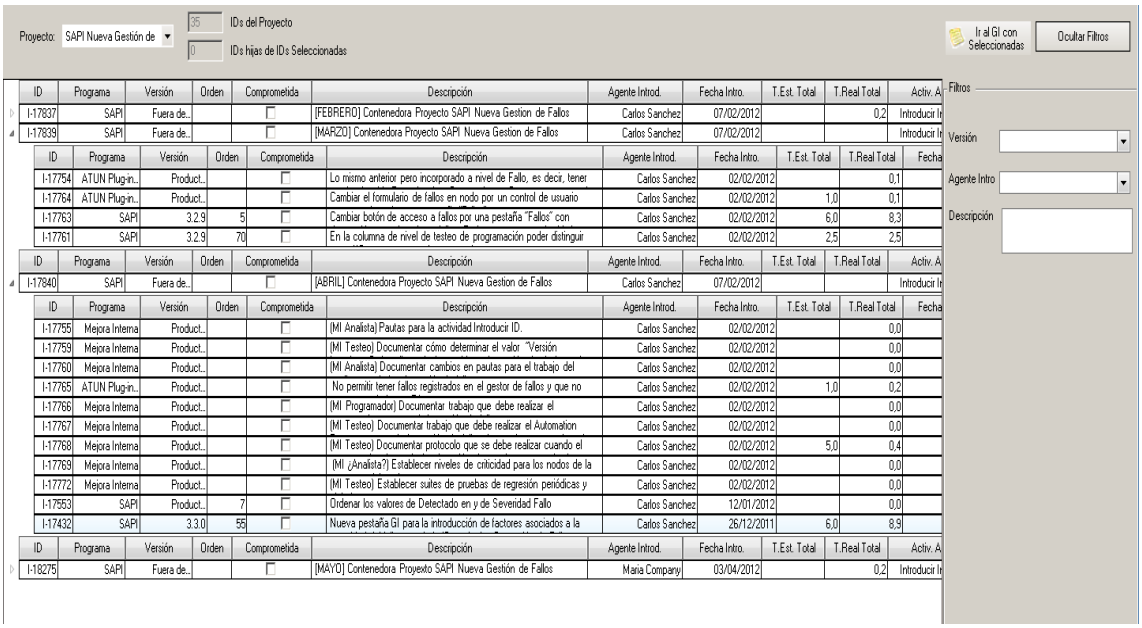


Ilustración 60. Gestor de Proyectos: "SAPI Nueva Gestión de fallos"

4.3. Visualización de Proyectos

En este capítulo se van a presentar dos técnicas para la visualización de proyectos, una de ellas TreeMaps es la que estará implementada en TUNE-UP y la otra grafos se propone como trabajo futuro.

4.3.1 TreeMaps

Los TreeMaps[18,19,20] es una técnica de visualización fue ideada a principios de los 90 por Ben Shneiderman, esta técnica de visualización permite visualizar datos masivos a partir de jerarquías. Las jerarquías viene representadas mediante cajas que se encuentran unas dentro de otras. Estas jerarquías de visualización permiten representar atributos a partir de sus tamaños y sus colores internos.

Los TreeMaps (Ilustración 61) nacen de la necesidad de pintar un disco duro lleno de manera fuera fácil encontrar aquellos archivos que ocupaban mas para que fueran eliminados, pero hoy en día se pueden encontrar para representar orden e importancia sobre objetos que están relacionados en relaciones de contención.

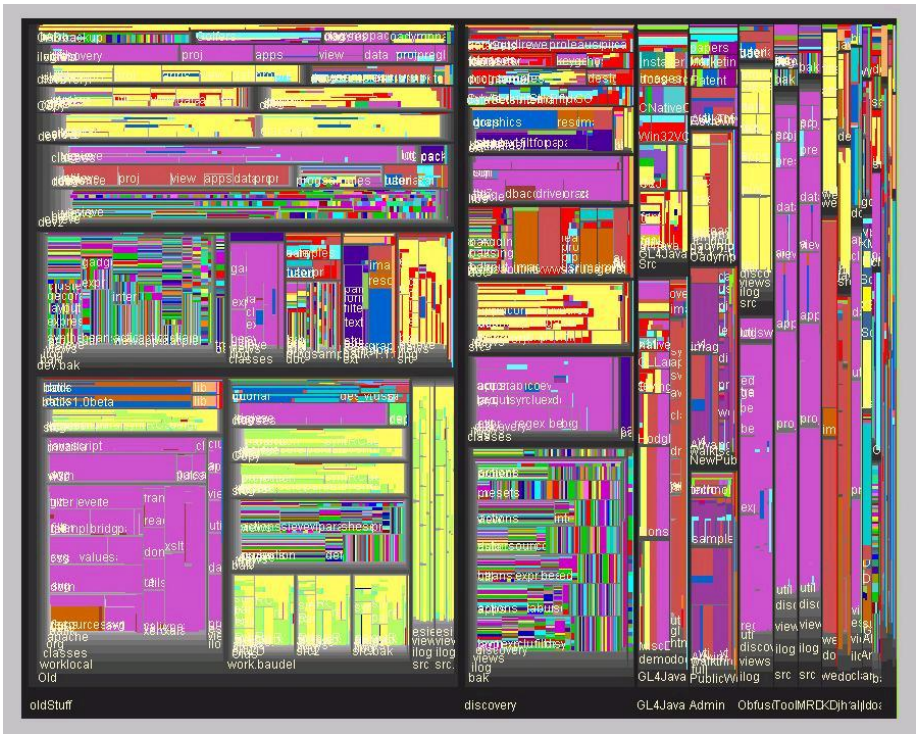


Ilustración 61.Ejemplo de TreeMap

La idea clave de los TreeMaps es que la jerarquía de los objetos está dada por el área del rectángulo que los representa: a mayor área, mayor importancia en la escala utilizada.

Para poder extraer una visualización en forma de TreeMap de los proyectos, vamos a utilizar el componente ultra chart que proporciona la libreria infragistics. Este componente necesita un datasource en forma de tabla y una serie de parámetros para que automáticamente nos dibuje TreeMap, por lo tanto la disposición de los objetos en el TreeMap el programador debe de despreocuparse ya que lo realiza de manera automática el componente.

Lo que debe de preocuparse el programador es de proporcionar de manera correcta los parámetros al componente.

Como hemos comentado anteriormente uno de los elementos que necesita el componente para dibujar el TreeMap es un data source en forma de tabla, pero esta tabla tiene que tener una serie de características para hacerlo más sencillo vamos a ir poco a poco viéndolo desde un ejemplo de una jerarquía sencilla.

Imaginemos que queremos representar en forma de TreeMap el siguiente árbol de objetos.

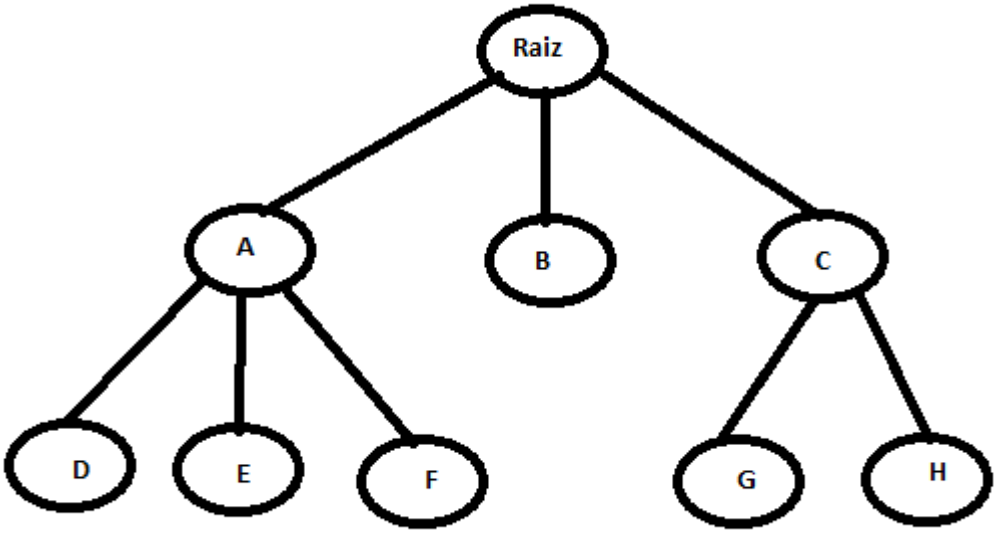


Ilustración 62. Ejemplo de jerarquía en forma de árbol

El data source en forma de tabla que debemos de insertar en el componente debe de poder representar de alguna manera esa jerarquía para mantenerla intacta a la hora de la visualización.

El componente propone una tabla donde las columnas muestren la jerarquía siendo la primera columna de la tabla padre de la segunda columna y así sucesivamente. De esta manera la representación en forma de tabla del árbol anterior sería el siguiente.

Columna 1	Columna 2	Columna 3
Raíz	A	D
Raíz	A	E
Raíz	A	F
Raíz	B	
Raíz	C	G
Raíz	C	H

Ilustración 63. Tabla que representa la jerarquía del árbol

En nuestro caso aplicado a los proyectos vemos que tendremos que implementar una tabla de manera dinámica, ya que no conoceremos el número de columnas que debe de tener nuestra tabla porque no conoceremos el nivel de profundidad. En la construcción de la tabla si nos fijamos bien cada fila de la tabla representa cada camino que hay desde la raíz del árbol hasta una hoja, por lo tanto deberemos de construir un algoritmo que dado un árbol nos de cada uno de los caminos que hay desde la raíz hasta las hojas. Este tipo de algoritmos existen desde que aparecieron los primeros arboles el problema es que si estos árboles tienen mucha profundidad el coste temporal de recorrer el árbol puede ser bastante grande.

Una vez establecido la construcción de la tabla es necesario añadir las columnas con los atributos que vamos a representar, en nuestro caso vamos a representar el tiempo estimado y el esfuerzo a priori, aquí se nos planteaba una duda de quien será esos atributos, del padre superior o de los hijos, por cuestiones lógicas los atributos serán de las hojas ya que un padre se medirá en función del tamaño de sus hijos. Por lo que los parámetros se añadirán en forma de columna a la tabla.

Columna 1	Columna 2	Columna 3	Esfuerzo a Priori	Tiempo estimado
Raíz	A	D	3	2
Raíz	A	E	4	3
Raíz	A	F	7	5
Raíz	B		1	0,5
Raíz	C	G	1	0,5
Raíz	C	H	2	1

Ilustración 64. Tabla de representación de jerarquía de árbol con columnas de atributos

Las columnas se añaden a la tabla en este caso al final debido a que el primer camino es de profundidad igual a la profundidad del árbol, pero es posible que estas columnas no se encuentren al final incluso que estén separadas, en ese caso además de proporcionarle la tabla al componente se le debe de proporcionar un vector con las posiciones de las columnas que representan la jerarquía, en este caso el si queremos omitir la raíz seria 2,3.

Otra de las cosas que debemos de indicarle al componente es la posición de la columna que representará el atributo que queremos pintar, en el componente solo podremos indicar uno de los atributos, la elección de uno u otro requerirá el repintado de TreeMap.

El resultado final utilizando la variable esfuerzo a priori sería la siguiente imagen.

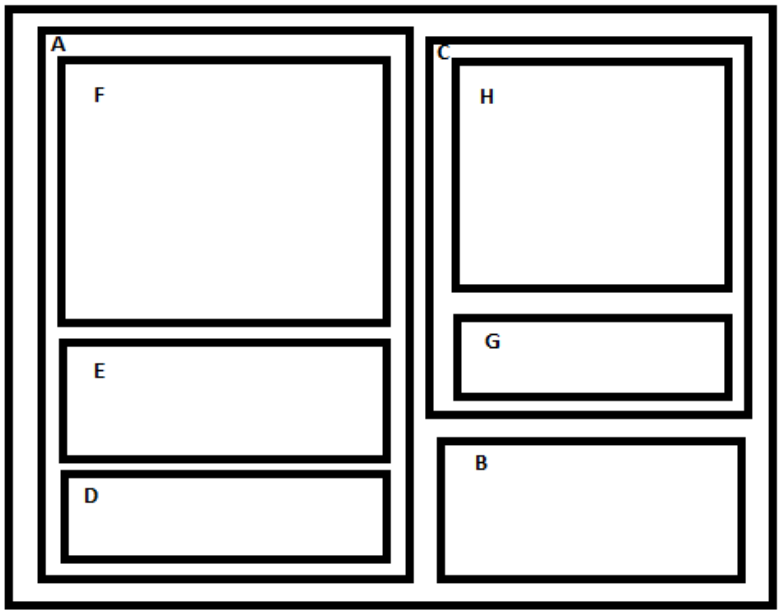


Ilustración 65. Ejemplo de TreeMap del Árbol Jerárquico

Para los proyectos dentro de TUNE-UP se ha creado un nuevo formulario para la visualización del TreeMap, este formulario será accesible a través del botón visualizar dentro del propio formulario de proyectos.

El nuevo formulario estará dividido en dos partes, una de ella contendrá un combo desde el cual podremos seleccionar las distintas variables que establezcan el tamaño de los objetos dentro del TreeMap, la otra parte será la representación del mismo.

Cuando se cambie de parámetro desde el combo esto requerirá de un nuevo repintado del TreeMap. A continuación se muestran una serie de ejemplos de visualizados en TreeMaps de algunos proyectos dentro de TUNE-UP.

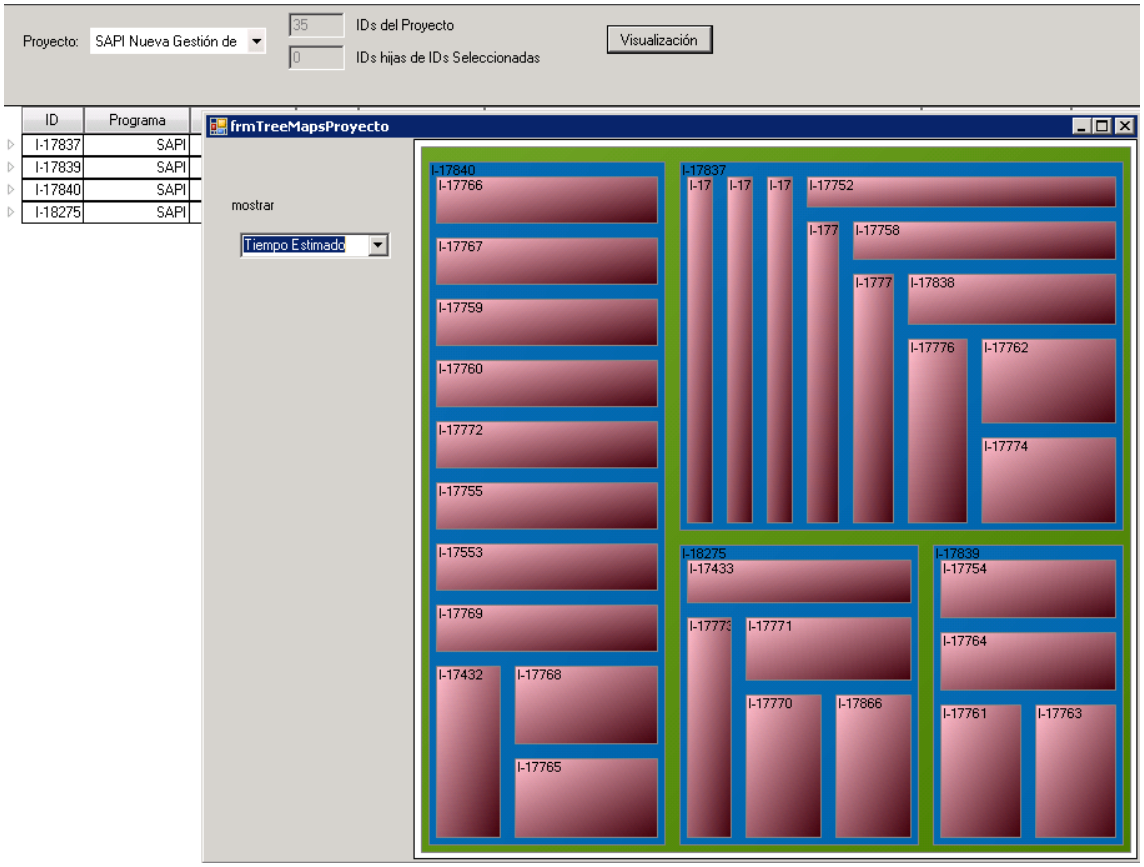


Ilustración 66 Proyecto Nueva Gestión de Fallos con Tiempo Estimado

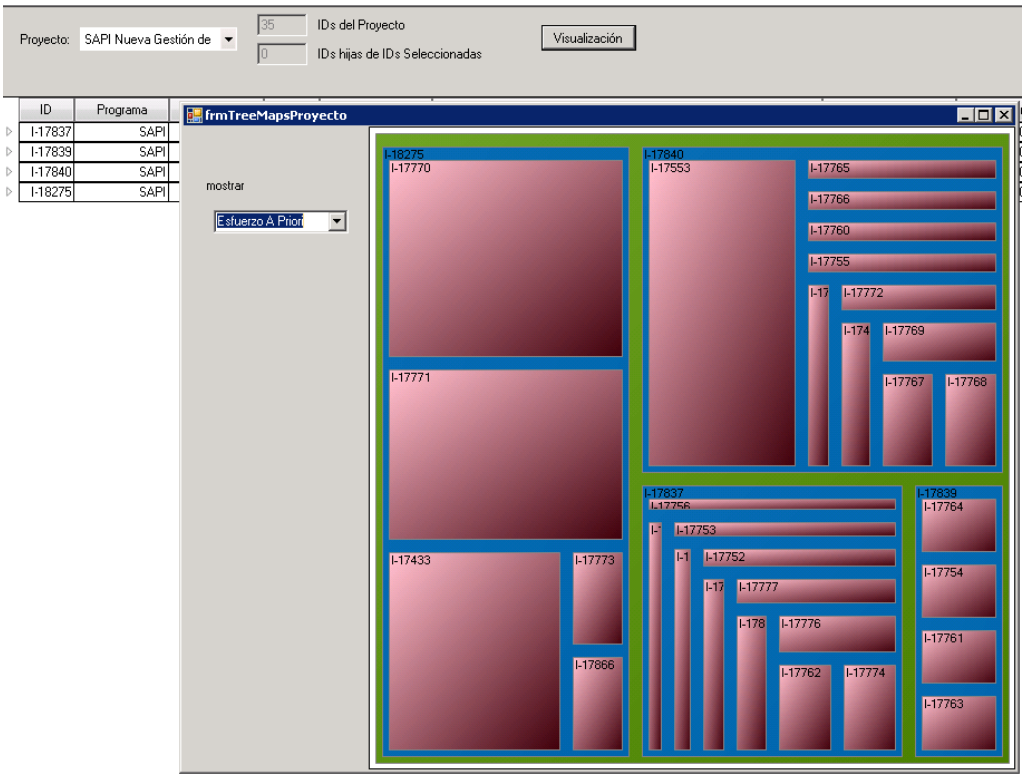


Ilustración 67 Proyecto Nueva Gestión de Fallos con Esfuerzo a Priori

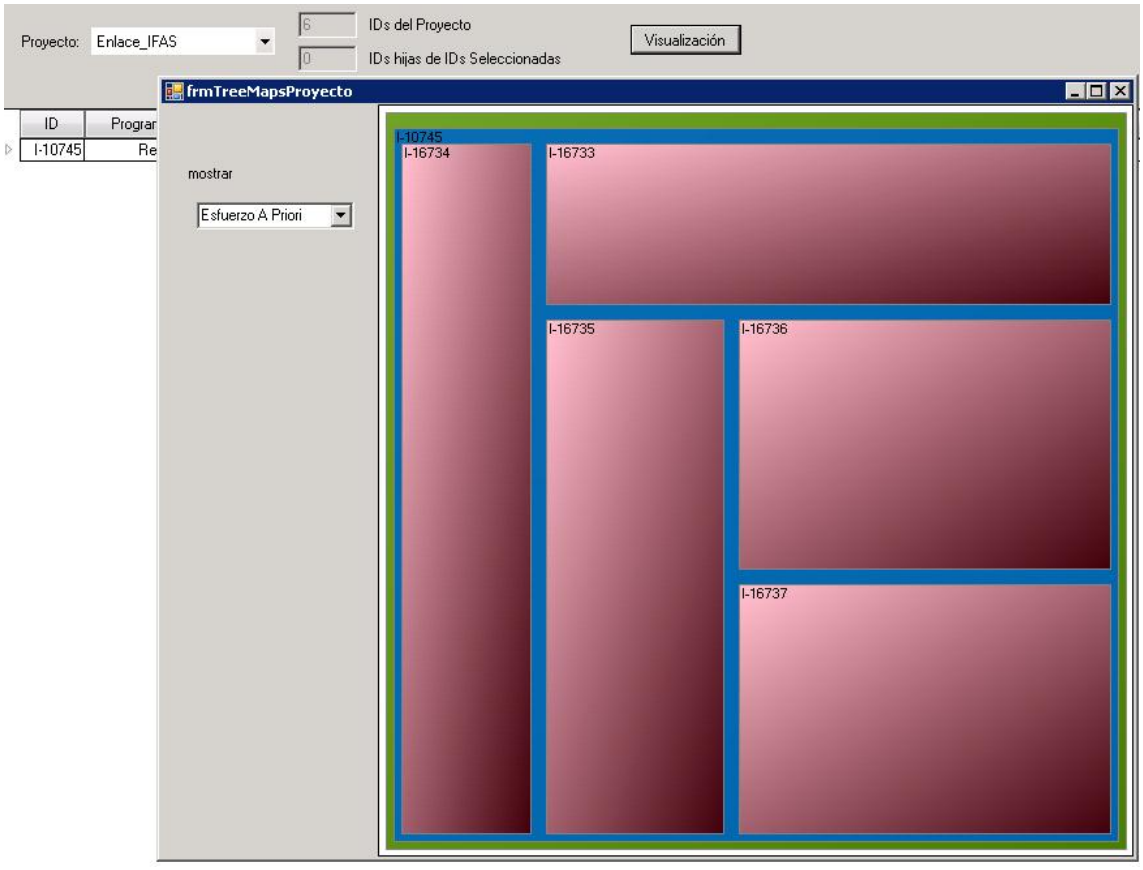


Ilustración 68 Proyecto Enlace con Esfuerzo a Priori

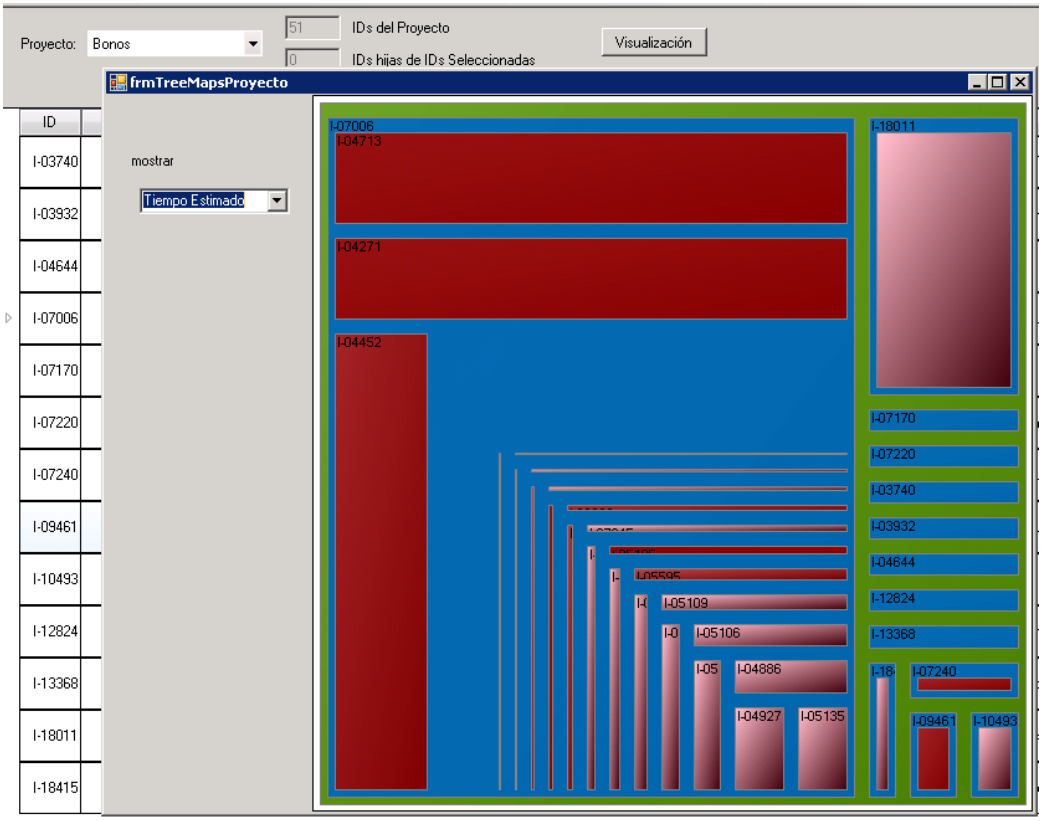


Ilustración 69 Proyecto Bonos con Tiempo Estimado

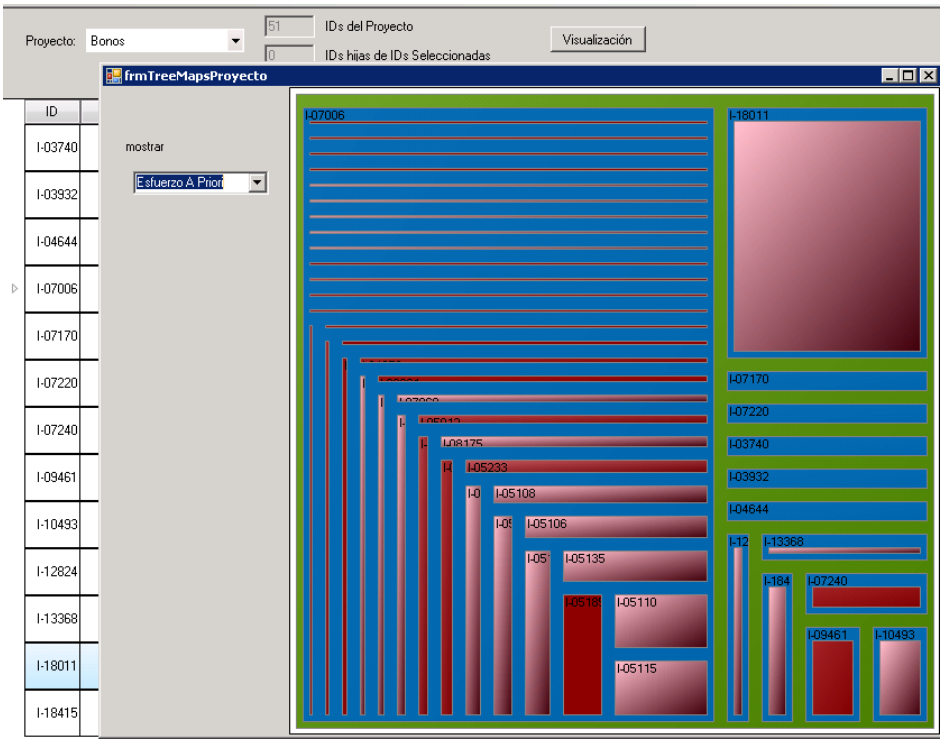


Ilustración 70 Proyecto Bonos con Esfuerzo a Priori

4.3.2 Grafos

Otra manera de visualización para la estructura del proyecto es mediante un grafo coloreado, este tipo de grafo nos permite visualizar datos masivos, es decir, la representación de muchos datos.

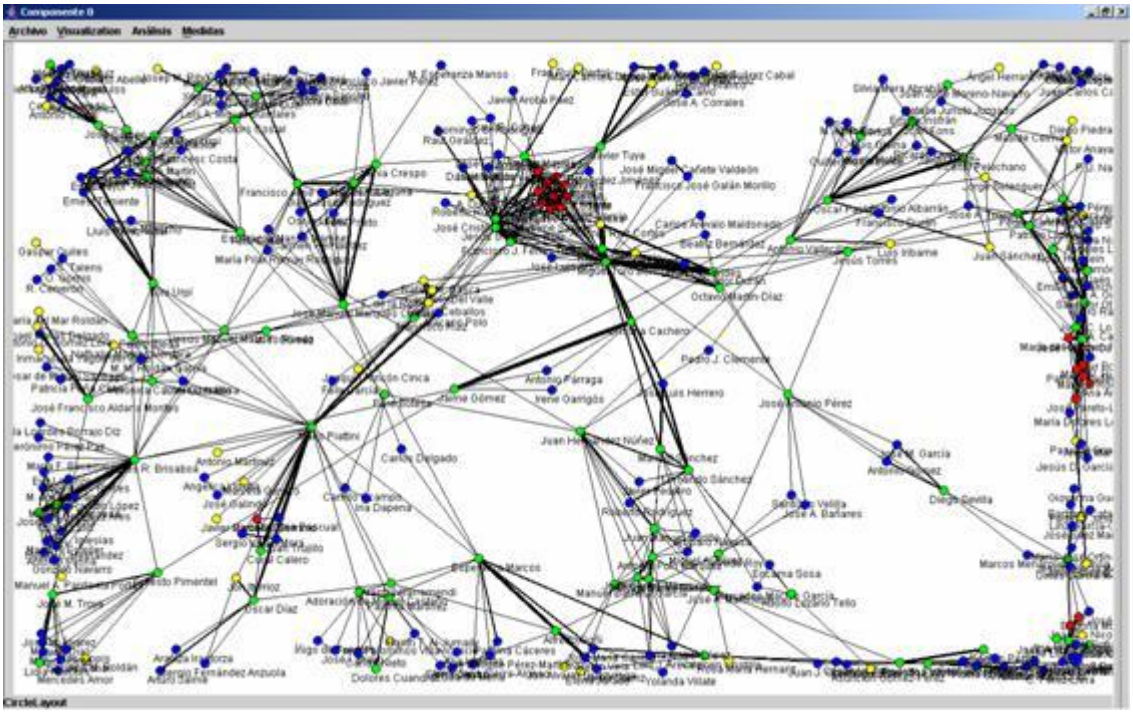


Ilustración 71. Ejemplo de Grafo Coloreado

Esto grafos tiene la peculiaridad de que los nodos puede tener un tipo de color y por lo tanto la agrupación entre ellos.

En nuestro caso los nodos serian WUs o grupos de WUs, los arcos relaciones entre WUs y el color nos serviría para distinguir el producto al que pertenece la WU.

Una de las diferencias entre un grafo y un TreeMap es en su capacidad de representar las relaciones, mientras que el TreeMap solo se pueden representar las relaciones verticales, es decir, relaciones del tipo composición (Padres-hijos), en los grafos además de la relaciones verticales se puede pueden representar relaciones horizontales como las relaciones de continuidad, relaciones de interés y otro tipo de relaciones horizontales.

Como hemos nombrado anteriormente se pueden expresar las relaciones horizontales esto se hace situando los nodos al mismo nivel y uniéndolos haciendo un arco sobre ellos. En nuestro caso tenemos bastantes tipos de relaciones horizontales por lo que dependiendo del tipo de arco de unión podemos diferenciar el tipo de relación.

Hay varios frameworks que nos pueden ayudar a dibujar los nodos y las relaciones entre los nodos, el problema de estos frameworks es la disposición de los nodos sobre la pantalla cuando el tamaño de los nodos es excesivo, esto es debido a que no existe un layout definido que permita este tipo de visualizaciones. Una posible solución seria el pintado por niveles, primero se visualizaría un nodo que al hacer pinchar sobre este, el nodo se expandiera en otros niveles y así sucesivamente.

Otra características que nos permiten estos frameworks es el uso de regiones para englobar distintos nodos, esto nos serviría dentro de los proyectos para establecer los plazos de finalización de las WUs dentro del proyecto.

Este tipo de visualización no esta implementado actualmente en TUNE-UP pero queda reflejado que seria bastante interesante por el hecho de la posibilidad de visualizar relaciones de tipo horizontales y por lo tanto se deja pendiente su implementación como parte de un trabajo futuro.

Capitulo 5. Conclusiones

Un producto software que no evoluciona y que no sufre una serie de transformaciones durante su ciclo de vida es un producto que acaba siendo sustituido por los clientes por otro producto que incorpore las mejoras.

En esta tesis se han abordado dos desafíos importantes dentro de un producto de gran envergadura; refactorización de requisitos y coordinación de proyectos.

En la tesis nos hemos enfrentado a dos desafíos que hacen evolucionar al producto por lo que hemos tenido que aplicar metodologías agiles para poder obtener un resultado de manera rápida y fácil y sin que con ello interfiramos en el resto del funcionamiento del producto.

Además del desarrollo hemos aplicado una nueva técnica para los inicios de un proyecto con gran éxito. que nos ha permitido coordinar correctamente varios grupos de trabajos sin ocasionar los problemas que normalmente derivan esta clase de proyectos. Esta nueva técnica ha sido implementada por los equipos y por lo tanto se ha incorporado la técnica dentro de la gestión de proyectos.

Otro tema que se ha tratado en la tesis es la visualización de datos masivos hemos visto dos tipos de herramientas diferentes que nos permiten ver ciertos atributos y cumplir el objetivo tener claro que WU es más prioritaria o más importante realizar de un simple vistazo.

Vista la utilidad de los TreeMaps para la representación de las incidencias que incluyen un proyecto, proponemos como trabajo futuro el uso de la visualización de TreeMaps para visualizar la estructura del producto en el gestor de requisitos(REM), asimismo, proponemos una representación en forma de grafo para visualizar los proyectos y la estructura del producto.

Los desafíos implementados se ha desarrollado en el contexto de una colaboración con la empresa Aphelion Soluciones Informáticas S.L, una pyme de unos 60 empleados que se encargan de desarrollar y vender un ERP para el sector sociosanitario. Actualmente la empresa usa la metodología TUNE-UP para la gestión de más de 15000 pruebas de aceptación.

Capitulo 6. Referencias

1. M. Company. Análisis de Impacto de Requisitos en un proceso de desarrollo centrado en Pruebas de Aceptación (2011).
2. Marante, M. Company, M. Letelier, P. Suarez, F. Gestión de requisitos basada en pruebas de aceptación: Test-Driven en su máxima expresión. XV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2010).
3. Company, M., Letelier, P., Marante, M., Suarez, F. Análisis de impacto en requisitos soportado de forma semi-automática y en un marco de desarrollo TDD basado en pruebas de aceptación. XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2011)
4. Marante, M., Company, M., Letelier, P. Seguimiento ágil de proyectos de desarrollo de software utilizando Gráficas Burn Down. XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2011)
5. Marante, M., Letelier, P., Suarez, F. *TUNE-UP: Seguimiento de proyectos software dirigido por la gestión de tiempos.* XIV Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2009). ISBN 978-84-692-4211-7 (pág. 57-68), Septiembre 2009.
6. Marante, M., Letelier, P., Suarez, F. *TUNE-UP: Un enfoque pragmático para la planificación y seguimiento de proyectos de desarrollo y mantenimiento de software*. I Congreso Iberoamericano SOCOTE – SOporte al COnocimiento con la TEcnología (SOCOTE 2009). ISBN 978-84-613-8585-0, Noviembre 2009.
7. Haugset, B., Hanssen, G., Automated Acceptance Testing: A Literature Review and an Industrial Case Study, Proc. of Agile 2008, pp. 27-38
8. Kroll P., Kruchten P., Booch G. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley Professional. 2003.
9. Beck K. Extreme Programming Explained: Embrace Change. Addison-Wesley. 2000.
10. Cohn, M. User Stories Applied for Agile Software Development, Person Education, Inc., 2004.

11. IEEE 830-1993, IEEE Recommended Practice for Software Requirements Specifications
12. Humphrey, Watts S. The Personal Software Process. Software Process Newsletter, Technical Council on Software Engineering, IEEE Computer Society, Volume 13, No. 1, Sept. 1994, pp SPN 1-3.
13. Allen, D. Getting Things Done: The Art of Stress-Free Productivity. Published in Penguin Books 2003. ISBN 0 14 20.0028 0
14. Covey, S. The Seven Habits of Highly Effective People. Published Simon & Schuster, Limited. Edition Softcover. ISBN 0743269519
15. Jeff Patton, jpatton@acm.org ,www.agileproductdesign.com
16. Bersoff, E. H. , henderson,V. D. ,and Siegel, S. G. Software Configuration Management. Prentice-Hall,1980
17. Bruls, Mark; Huizing, Kees; van Wijk, Jarke J. (2000), "Squarified treemaps", in de Leeuw, W.; van Liere, R., *Data Visualization 2000: Proc. Joint Eurographics and IEEE TCVG Symp. on Visualization*, Springer-Verlag, pp. 33–42.
18. Shneiderman, Ben; Plaisant, Catherine (June 25, 2009). "Treemaps for space-constrained visualization of hierarchies". Retrieved February 23, 2010.
19. Roel Vliegen; Erik-Jan van der Linden and Jarke J. vanWijk. "Visualizing Business Data with Generalized Treemaps" (PDF). Retrieved February 24, 2010.
20. Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide) - Fourth Edition, 1987
21. Dennis Lock. Fundamentos de la gestión de proyectos; 2003; Ed. Aenor