Revista
Española de

Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 3, No. 3, diciembre, 2007

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

Publicado por la Asociación de Técnicos en Informática

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

Contenidos

REICIS

Estrategia de gestión de las pruebas funcionales en el Centro de Ensayos de Software

Beatriz Pérez Lamancha
Centro de Ensayos de Software
bperez@fing.edu.uy

Abstract

This article presents a strategy for managing the functional testing of a software product. The strategy defines testing scope and agenda based on the product risk analysis, combining test cases design with exploratory testing. Test cycles are defined according to the product development plan and an overall test plan is prepared based on its functionalities, which is reviewed and refined before each cycle starts. Exploratory testing has a fundamental role in this approach. On the one hand, it helps to mitigate possible errors in the product risk analysis as leaving out important business functionalities. The product risk analysis must be reviewed when a critical bug for the business is found during an exploratory testing session, which was not detected by the execution of designed test cases. On the other hand, exploratory testing complements the test cases when the available time is not enough to design test cases to cover all the functionalities in the test cycle.

KeyWords: Software engineering, testing, test management, functional testing.

Resumen

Se presenta en este artículo una estrategia para la gestión de las pruebas funcionales de un producto de software. La estrategia define el alcance y la agenda de las pruebas a partir del análisis de riesgo del producto, combinando los casos de prueba con diseño previo y el testing exploratorio. Se definen los ciclos de prueba en función del plan de desarrollo del producto y se elabora una planificación global a partir de sus funcionalidades, la que es revisada y refinada al comenzar cada ciclo de prueba. El testing exploratorio cumple un papel fundamental en la estrategia. Por un lado, ayuda a mitigar la posibilidad de equivocarse al realizar el análisis de riesgo del producto, dejando de lado funcionalidades importantes para el negocio. Si en una sesión de testing exploratorio se encuentra un incidente crítico para el negocio que no fue detectado por los casos de prueba diseñados, el análisis de riesgo debe ser revisado. Por otro lado, complementa la prueba con diseño previo cuando no se dispone del tiempo suficiente en un ciclo como para generar los casos de prueba que cubran las funcionalidades requeridas.

Palabras clave: Ingeniería de software, pruebas, gestión de las pruebas, Pruebas Funcionales.

1. Introducción

El objetivo de las pruebas funcionales es validar si el comportamiento observado del software cumple o no con sus especificaciones. La prueba exhaustiva del producto requiere ejercitar todos los caminos posibles del mismo, incluso para un programa pequeño, el tiempo requerido para esto es excesivo. Esto impacta directamente en la economía de las pruebas, ya que se deberán realizar suposiciones sobre el comportamiento del programa y la forma en que se diseñan los casos de prueba para el mismo. El objetivo es maximizar la producción de las pruebas, esto es, maximizar el número de los errores encontrados por un número finito de los casos de prueba [1]. La técnica con que se seleccionan los casos de prueba es uno de los principales supuestos a definir.

Existen distintos procesos y metodologías para las pruebas funcionales de software, agrupan en general las actividades referentes a las pruebas en las etapas de planificación, diseño y de ejecución de las pruebas. Durante la planificación de las pruebas se decide qué se probará y con qué profundidad, en la etapa de diseño, la especificación de requisitos se analiza para derivar los casos de prueba y en la etapa de ejecución se ejecutan los casos de prueba diseñados previamente, se compara el resultado real con el esperado y se reportan los resultados.

Se presenta en este trabajo una estrategia para la gestión de las pruebas funcionales que define el alcance y la agenda de las pruebas del proyecto en función del análisis de riesgo del producto, combinando los casos de prueba derivados utilizando técnicas de caja negra y el testing exploratorio. De esta forma se retroalimentan los casos de pruebas diseñados con los resultados del testing exploratorio. Esta estrategia de gestión es la utilizada para realizar servicios de prueba independiente en el Centro de Ensayos de Software [1] (CES).

El resto del artículo está organizado como sigue. La siguiente sección describe brevemente los principales conceptos referentes a las pruebas funcionales. La sección 3 presenta el Centro de Ensayos de Software y el proceso de pruebas funcionales que sigue en sus proyectos. La sección 4 describe la estrategia de planificación de las pruebas funcionales. Finalmente, la sección 5 presenta las conclusiones y trabajo futuro.

[1] Centro de Ensayos de Software, Montevideo, Uruguay. http://www.ces.com.uy

2. Pruebas funcionales

En esta sección se definen los principales términos referidos en este trabajo. Se define prueba funcional, prueba de regresión y *testing* exploratorio.

La prueba funcional es conocida también como basado en la especificación o de caja negra. El objetivo de la prueba funcional es validar si el comportamiento observado del software cumple o no con sus especificaciones. La prueba funcional toma el punto de vista del usuario [2]. Las funciones son probadas ingresando las entradas y examinando las salidas, la estructura interna del programa raramente es considerada [3]. Para realizar pruebas funcionales, la especificación se analiza para derivar los casos de prueba. Técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto y conjetura de errores son especialmente pertinentes para las pruebas funcionales. Se deben considerar condiciones inválidas e inesperadas de la entrada y tener en cuenta que la definición del resultado esperado es una parte vital de un caso de la prueba. El propósito de la prueba funcional es mostrar discrepancias con la especificación y no demostrar que el programa cumple con su especificación [1].

Las pruebas de regresión tienen como objetivo verificar que no ocurrió una regresión en la calidad del producto luego de una modificación, asegurando que los cambios no introducen un comportamiento no deseado o errores adicionales. Implican la reejecución de alguna o todas las pruebas realizadas anteriormente [4].

El término "*testing* exploratorio" fue introducido por Cem Kaner [5], se trata de ejecutar las pruebas a medida que se piensa en ellas, utilizando muy poco tiempo en preparar o explicar las pruebas, confiando en los instintos. El *testing* exploratorio se define como el aprendizaje, el diseño de casos de prueba y la ejecución de las pruebas en forma simultánea. En otras palabras, es cualquier prueba en la cual quien prueba controla activamente el diseño de las pruebas mientras las pruebas se ejecutan, y utiliza la información obtenida mientras prueba para diseñar nuevas y mejores pruebas [5]. En el *testing* exploratorio siempre se debe tomar nota de lo que se hizo y lo que sucedió [3]. Los resultados obtenidos no son necesariamente diferentes de aquellos obtenidos de la prueba con diseño previo y ambos enfoques para las pruebas son compatibles [5].

El *testing* exploratorio puede ser realizado en cualquier situación donde no sea obvio cual es la próxima prueba que se debe realizar. También cuando se requiere obtener

retroalimentación rápida de cierto producto o funcionalidad, se necesita aprender el producto rápidamente, se quiere investigar y aislar un defecto en particular, se quiere investigar el estado de un riesgo particular, o se quiere evaluar la necesidad de diseñar pruebas para determinada área. Una estrategia básica para realizar *testing* exploratorio es tener un plan de ataque general, pero permitirse desviarse de él por periodos cortos de tiempo. Cem Kaner llama a esto el principio del "tour bus", las personas bajan del bus y conocen los alrededores. La clave es no perderse el tour entero [5]. En general, el *testing* solamente exploratorio puede funcionar para *testers* con mucha experiencia. Como ventaja se encuentra que es barato y rápido, como contra que no se tienen medidas de cubrimiento y no deja documentación [4].

3. Centro de Ensayos de Software

El Centro de Ensayos de Software (CES) es un emprendimiento conjunto de la Universidad de la República de Uruguay (UdelaR) y de la Cámara Uruguaya de Tecnologías de la Información (CUTI), entidad que agrupa a la mayoría de las empresas productoras de software de Uruguay. Tiene por objetivo brindar servicios especializados de testing a la industria de tecnologías de la información (TI), para mejorar su capacidad productiva en cuanto a calidad, diversidad de plataformas e innovación de sus productos.

Los servicios que ofrece el CES incluyen

- Servicios de prueba independiente: Planificar, diseñar, coordinar y ejecutar pruebas de productos de software de manera efectiva y controlada, definiendo claramente el contexto y los objetivos.
- Consultoría: Asesorar a las organizaciones en la mejora de los procesos de prueba, definición de estrategias y automatización de las pruebas. Colaborar en la creación y consolidación de sus áreas de prueba.
- Capacitación: Elaborar e impartir programas de capacitación en la disciplina de testing según las necesidades de cada organización.

El CES se compone de dos laboratorios: el Laboratorio de Testing Funcional enfocado en la evaluación de productos desde el punto de vista funcional y el Laboratorio de Ensayos de Plataformas, donde se realizan pruebas de desempeño y se asiste a la

industria para resolver problemas de funcionamiento en arquitecturas de hardware y software complejas.

En la Figura 1 se muestra la cantidad de proyectos en los que ha trabajado el CES desde su creación, distinguiendo por tipo de proyecto hasta junio de 2007.

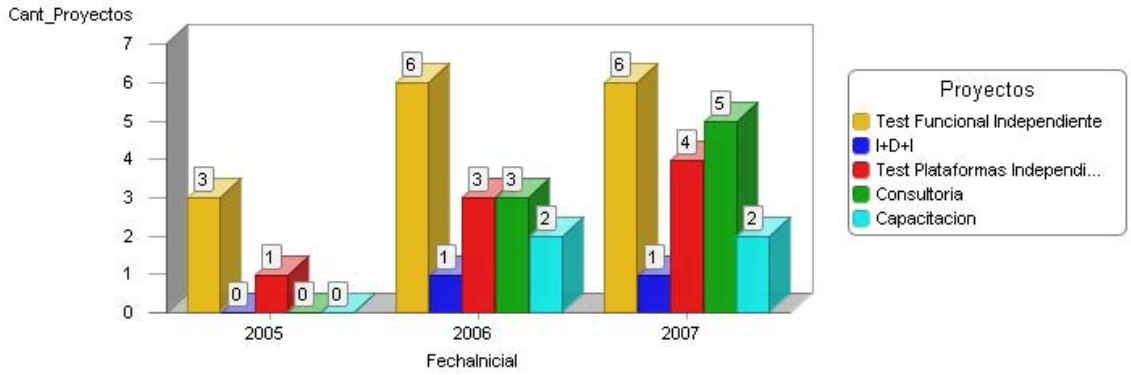

Figura 1. Cantidad de Proyectos por tipo y año

El monto total de ingreso de los distintos proyectos puede clasificarse de la siguiente manera:

- un 89% corresponde a organizaciones privadas y un 11% a organizaciones públicas
- un 33% corresponde a pruebas funcionales independientes, un 33,6% corresponde a consultoría, un 30% a ensayos de plataformas y un 3,4% a capacitación.
- Un 90% son proyectos en Uruguay y un 10% son proyectos en el extranjero
- Un 70% de los proyectos corresponden a organizaciones que producen software, un 28% a organizaciones que consumen software y un 2% a organizaciones consultoras

Los proyectos de consultoría han sido en su gran mayoría para ayudar a formar el área de pruebas en las organizaciones y/o mejorar su metodología de pruebas. La estrategia que se presenta en este artículo fue concebida inicialmente para las pruebas funcionales independientes, donde un equipo del CES es contratado para probar un producto de software dentro de un intervalo de tiempo definido. Dicha estrategia luego también ha sido utilizada en consultorías, donde las organizaciones adoptan esta estrategia para probar sus

propios productos, la estrategia de gestión se adapta muy bien a este contexto. Las áreas de prueba internas de las organizaciones pueden utilizar esta estrategia para planificar sus pruebas funcionales y mantener las actividades de pruebas controladas.

3.1. Metodología para las pruebas utilizada en el CES

ProTest [6] es el proceso para pruebas funcionales de productos de software utilizado en el CES. El proceso tiene cuatro etapas: Estudio Preliminar, Planificación, Ciclo de Prueba y Evaluación del Proyecto, las cuales se muestran en la Figura 2.

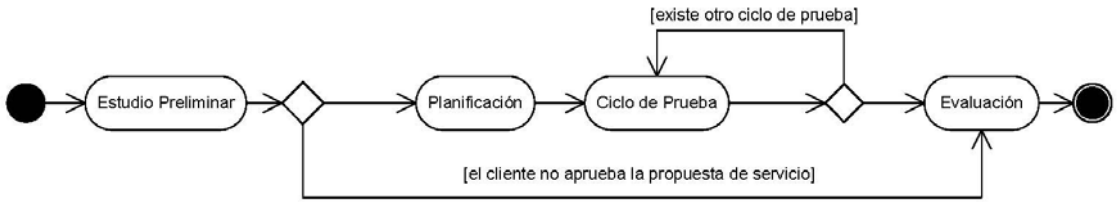

Figura 2. Etapas de ProTest

Los principales objetivos de cada etapa son:

- Estudio Preliminar: En esta etapa, se estudian las principales funcionalidades del producto, con el objetivo de definir el alcance de las pruebas y un primer cronograma de los ciclos de prueba. A partir de estos datos se realiza la propuesta de servicio de prueba. Si el cliente aprueba la propuesta de servicio de prueba, se sigue con la etapa de Planificación, en caso contrario se pasa a la etapa de Evaluación, donde se analiza cuales fueron los problemas en este proyecto y se archiva para su consulta en futuros proyectos.
- Planificación: El objetivo de esta etapa es planificar el proyecto de prueba. Se definen los ciclos de prueba y las funcionalidades a probar en cada ciclo en función del análisis de riesgo del producto. Se genera el Plan de Pruebas que resume toda la información del proyecto de prueba y las decisiones tomadas durante la etapa de Planificación.
- Ciclo de Prueba: El objetivo de esta etapa es generar y ejecutar las pruebas para una versión determinada del producto. El proyecto de prueba es guiado por los ciclos de prueba, cada ciclo de prueba está asociado a una versión del producto a probar.

- Evaluación: Esta etapa tiene como objetivo conocer el grado de satisfacción del cliente, realizar el informe final, evaluar el proceso de prueba para su mejora y almacenar los elementos del proyecto de prueba para su uso en proyectos posteriores.

La etapa Ciclo de Prueba se divide a su vez en cuatro sub-etapas: Seguimiento del Ciclo, Configuración del Entorno, Diseño de las Pruebas y Ejecución de las Pruebas, las cuales se muestran en la Figura 3. A continuación se describen los objetivos de cada sub-etapa dentro del Ciclo de Prueba:

- Seguimiento del Ciclo: El objetivo es realizar el seguimiento y control del ciclo de prueba. La planificación realizada al principio del proyecto es revisada al comenzar cada ciclo de prueba, se planifican en detalle las tareas para el ciclo y se ajusta la planificación según las estimaciones y posteriores mediciones realizadas en los ciclos anteriores.
- Configuración del Entorno: El objetivo es configurar el ambiente de pruebas, separándolo del ambiente de desarrollo, instalar las herramientas necesarias y el software a probar en la versión correspondiente a cada ciclo de prueba.
- Diseño de las Pruebas: Consiste en el diseño de los casos de prueba a partir de la especificación del producto.
- Ejecución de las Pruebas: El objetivo de esta etapa es contrastar el comportamiento esperado del software con su comportamiento real, analizar las diferencias y reportar los resultados.

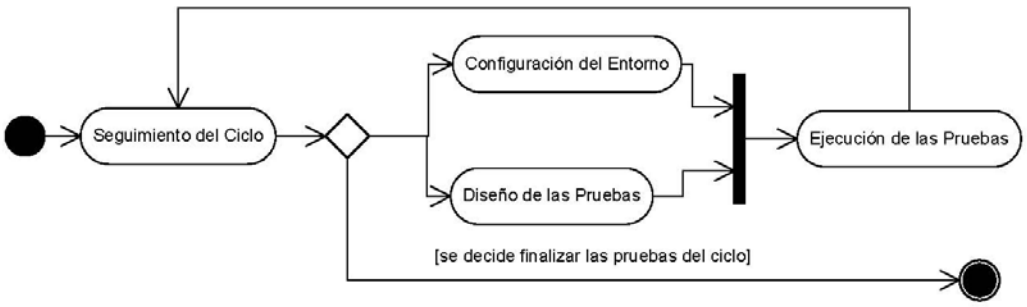

Figura 3. Ciclo de Prueba

Los ciclos de prueba se solapan en el tiempo, mientras se ejecutan las pruebas para un ciclo, se puede al mismo tiempo estar diseñando las pruebas del ciclo siguiente. En cada

ciclo se revisan solamente las especificaciones de las funcionalidades que serán probadas en ese ciclo, con el objetivo de poder diseñar las pruebas que serán ejecutadas en ese ciclo. En [7] y [6] pueden consultarse casos de estudio donde se puso en práctica esta metodología.

4. Cuestiones relativas a la adaptación del proceso software.

En esta sección se presenta la estrategia para la gestión de las pruebas funcionales de un producto de software utilizada en el Centro de Ensayos de Software. Las principales características de la estrategia son que se basa en realizar un estudio de los riesgos del producto que permita definir el alcance para las pruebas, define los ciclos de prueba que se realizarán del producto en función del plan de desarrollo del producto y usa un enfoque iterativo para la planificación de las pruebas, donde se define una planificación general a nivel macro de las funcionalidades a probar en cada ciclo, la que se revisa y refina al comenzar cada ciclo de prueba. Esta metodología fue concebida para realizar pruebas manuales, pero puede ser adaptada para proyectos de automatización de las pruebas, donde la planificación de los casos de prueba a automatizar es realizada siguiendo esta estrategia, como se describe en [8].

4.1. Prueba basada en los riesgos del producto

El enfoque basado en los riesgos tiene tres pasos: primero confeccionar una lista priorizada de riesgos, luego realizar pruebas que exploran cada riesgo, y por último, cuando un riesgo se mitiga y emergen nuevos, se debe ajustar el esfuerzo de la prueba [9]. Un riesgo es la probabilidad de que algo no deseado ocurra. La magnitud de un riesgo es proporcional a la probabilidad y el impacto del problema. A mayor probabilidad de ocurrencia y mayor impacto, mayor es el riesgo asociado a ese problema. Existen dos enfoques heurísticos para el análisis de riesgo, uno "desde adentro hacia fuera" y otro "desde afuera hacia adentro". Son acercamientos complementarios, cada uno con sus fortalezas. El enfoque de adentro hacia fuera pregunta "¿qué riesgos se asocian a esta funcionalidad?", mientras que el enfoque de afuera hacia adentro es el opuesto "¿qué funcionalidades se asocian a esta clase de riesgo?".

El enfoque "desde afuera hacia adentro" estudia una lista de riesgos potenciales y se realiza la correspondencia con los detalles del producto. Con este enfoque, se consulta una lista predefinida de riesgos y se determina si aplican [10].

El enfoque usado en este artículo para el estudio de los riesgos del producto es el denominado "desde adentro hacia fuera", donde se identifican las funcionalidades del producto y los riesgos asociados a cada una, por ejemplo, tendrán mayor riesgo aquellas funcionalidades que en caso de fallar tienen las consecuencias más serias para el negocio o aquellas que tienen mayor frecuencia de uso, ya que si una parte del sistema es usada frecuentemente y tiene un error, su uso frecuente probablemente hará aparecer la falla [9].

4.2. Ciclo de prueba

Durante el ciclo de vida de un producto, sin importar cual sea el proceso de desarrollo, se van generando distintas versiones de la aplicación. Las actividades de la prueba se realizan para una determinada versión del producto, sobre la cual se ejecutan las pruebas y se reportan los incidentes encontrados. Las pruebas que serán ejecutadas sobre una versión son planificadas con anticipación y deberían ser ejecutadas, a menos que las prioridades cambien.

En un ciclo de prueba se puede ejecutar una, alguna o todas las pruebas planificadas para el producto [4]. Uno de los principales desafíos desde el punto de vista de la prueba independiente es estimar cuantos ciclos de prueba se requieren, ya que no todas las versiones que genera desarrollo llegan a ser probadas por el equipo de prueba, entre dos ciclos de prueba podrían existir más de dos versiones del producto generadas por el equipo de desarrollo.

4.3. Casos de estudio: experiencias en organizaciones reales

El alcance de las pruebas indica qué funcionalidades se van a probar y cuales no. Para poder definir el alcance, se divide el sistema en módulos, componentes o subsistemas, no todos los componentes serán probados con la misma importancia y pueden existir componentes que queden fuera del alcance de las pruebas. Cada componente agrupa varias funcionalidades, se dividen las funcionalidades hasta un nivel en el que sea posible definir el alcance. Luego de esto, se analizan las funcionalidades, dando como resultado un Inventario de Pruebas. El inventario es una lista de las funcionalidades del producto de software. Para cada funcionalidad se asigna:

- Identificador: Referencia única a la especificación de requerimientos, donde se encuentra la descripción detallada del mismo.
- Nombre: Descripción breve de la funcionalidad a probar.
- Prioridad: Indica la prioridad que tiene esa funcionalidad para las pruebas. Los valores posibles son: Alta, Media y Baja.

Estos valores se obtienen de utilizar el enfoque basado en los riesgos del producto de la sección 3.1. Se tomará como ejemplo un producto con 10 funcionalidades, se asume que el producto será construido en forma incremental y que se realizarán tres liberaciones internas desde desarrollo al equipo de pruebas antes de ser liberado el producto al ambiente de producción del cliente. En la Tabla 1 se muestra un ejemplo de la priorización realizada para dicho producto.

Inventario de Pruebas		
Identificador	**Nombre**	**Prioridad**
1	Funcionalidad 1	Media
2	Funcionalidad 2	Alta
3	Funcionalidad 3	Baja
4	Funcionalidad 4	Alta
5	Funcionalidad 5	Media
6	Funcionalidad 6	Baja
7	Funcionalidad 7	Alta
8	Funcionalidad 8	Media
9	Funcionalidad 9	Media
10	Funcionalidad 10	Baja

Tabla 1. Inventario de Pruebas

A partir de la estimación del esfuerzo de probar cada funcionalidad, el plan de desarrollo del producto y la fecha prevista para instalar el producto en el ambiente de producción, se define el alcance para las pruebas. El inventario permite recortar en forma ordenada el alcance, ya que es muy probable que las funcionalidades de prioridad baja sean las primeras en quedar fuera del alcance de las pruebas. La definición del alcance de las pruebas se realiza en reuniones donde participan los representantes del cliente, el gerente de proyecto, el líder de desarrollo y el líder de pruebas. Con un primer alcance para las pruebas definido, el próximo paso es la planificación de las pruebas funcionales en el tiempo, como se describe en la próxima sección.

4.4. Planificación de las pruebas funcionales

Para definir el cronograma de las pruebas a lo largo del proyecto, es necesario conocer el plan de desarrollo que contiene el cronograma de las versiones del producto que se

generarán. Se requiere conocer las fechas en que desarrollo tiene planificado liberar las versiones del producto al equipo de prueba y qué funcionalidades incorpora cada nueva versión. En la Tabla se muestra un ejemplo de cómo podría ser un plan de desarrollo del producto cuyo inventario es el de la Tabla 1, donde se tiene previsto construir el producto incrementalmente, liberando tres versiones intermedias al equipo de pruebas.

A partir de esta información el siguiente paso es estimar cuántos ciclos de prueba de la aplicación se van a realizar. Siguiendo con el ejemplo, se podría definir para el producto de la Tabla 1, que se va a realizar un ciclo de prueba por cada versión del producto y que finalmente se realizará un último ciclo de prueba previo a la instalación en producción para asegurarse que los incidentes encontrados en el último ciclo de prueba fueron solucionados, como se muestra en la Figura 4. En este caso, las fechas “Fecha 4” y “Fecha 5” deben ser acordadas con el equipo de desarrollo y el cliente. Se negocia con los desarrolladores el tiempo que necesitan para realizar las correcciones y generar la nueva versión a probar.

Plan de Desarrollo		
Versión	**Fecha**	**Id. De Funcionalidades que incluye el ejecutable**
1	Fecha 1	1,2,3,4
2	Fecha 2	1,2,3,4,5,6,7
3	Fecha 3	1,2,3,4,5,6,7,8,9,10

Tabla 2. Plan de Desarrollo

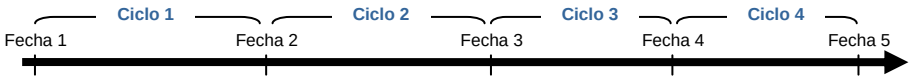

Figura 4. Ciclos de Prueba en el Tiempo

Una vez definidos los ciclos, las siguientes preguntas a responder, son: ¿Qué probar en cada ciclo? ¿Con qué profundidad se realizarán las pruebas de cada ciclo?. A partir de las funcionalidades del Inventario de Prueba de la Tabla 1, se refina cada componente, definiendo las funcionalidades en detalle. Se realiza nuevamente el análisis de riesgo para las nuevas funcionalidades. Puede ocurrir que un grupo de funcionalidades al que se le asignó prioridad alta, al ser dividido en varias funcionalidades, algunas de ellas sean de prioridad alta, otras de prioridad media y otras queden fuera del alcance de las pruebas. En el ejemplo de la Tabla 1, podría ocurrir que la funcionalidad 2, que tiene prioridad “Alta”, al refinarla, resulte en cuatro funcionalidades, como se muestra en la Tabla 2.

Con el inventario refinado, se definen las prioridades para las pruebas de cada ciclo. Del Plan de Desarrollo de la Tabla , surge que para el ciclo 1 sólo se contará con las funcionalidades 1, 2, 3 y 4 del producto. Del inventario de pruebas de la Tabla 2 surge que el orden de prioridad para las pruebas de dichas funcionalidades es: Funcionalidad 2.1, Funcionalidad 2.4 y Funcionalidad 4 con prioridad Alta, Funcionalidad 1 y Funcionalidad 2.3 con prioridades Media y Funcionalidad 1y Funcionalidad 2.2 con prioridad Baja. Ordenado por prioridad las funcionalidades, se obtiene para cada ciclo qué es lo más importante a probar. En la Figura 5 se muestra el orden de las funcionalidades en cada ciclo.

Además de planificar el diseño de las pruebas, se debe planificar el testing exploratorio que se realizará en cada ciclo de prueba. En general, el tiempo con que se cuenta entre un ciclo de prueba y el siguiente no es suficiente como para diseñar todos los casos de prueba de las funcionalidades de ese ciclo. Una posibilidad es complementar el diseño de las pruebas con el *testing* exploratorio. De esta manera, usando el análisis de riesgo del producto, se podría planificar el diseño de las pruebas de las funcionalidades de prioridad alta en cada ciclo y que las funcionalidades de prioridad media y baja, sean exploradas en sesiones de *testing* exploratorio.

Inventario de Pruebas			
Identificador	**Nombre**	**Funcionalidad en Detalle**	**Prioridad**
1	Funcionalidad 1	Funcionalidad 1	Media
2	Funcionalidad 2		Alta
2.1		Funcionalidad 2.1	Alta
2.2		Funcionalidad 2.2	Baja
2.3		Funcionalidad 2.3	Media
2.4		Funcionalidad 2.4	Alta
3	Funcionalidad 3	Funcionalidad 3	Baja
4	Funcionalidad 4	Funcionalidad 4	Alta
5	Funcionalidad 5	Funcionalidad 5	Media
6	Funcionalidad 6	Funcionalidad 6	Baja
7	Funcionalidad 7	Funcionalidad 7	Alta
8	Funcionalidad 8	Funcionalidad 8	Media
9	Funcionalidad 9	Funcionalidad 9	Media
10	Funcionalidad 10	Funcionalidad 10	Baja

Tabla 2. Inventario de Pruebas refinado

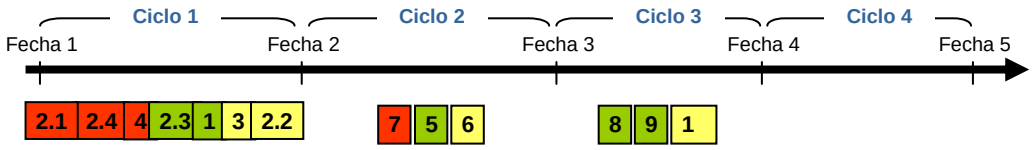

Figura 5. Prioridad de las funcionalidades en cada ciclo

4.5. Planificación de cada ciclo de prueba

Al comienzo del proyecto, se planifican las pruebas a realizar del producto en cada versión que se genere del mismo. Esta planificación debe ser revisada al comenzar cada nuevo ciclo de prueba, ya que los supuestos sobre los que se definió la planificación probablemente hayan cambiado. Por ejemplo, al transcurrir el tiempo pueden cambiar las prioridades de las pruebas debido a cambios en las prioridades del negocio o a la confianza adquirida en el producto como resultado de la realización de las pruebas en ciclos anteriores [4]

También en cada ciclo se deben planificar las pruebas de regresión. Al obtener una nueva versión donde se corrigieron incidentes, se deben ejecutar nuevamente los casos de prueba que encontraron esos incidentes. Como a priori no se conoce cuales ni cuantos serán esos incidentes, al comenzar cada ciclo, deben ser consideradas las pruebas de regresión.

5. Conclusiones

Se ha presentado una estrategia para la gestión de las pruebas funcionales, basada en la importancia de las funcionalidades a probar a través de un análisis de riesgo del producto y combinando los enfoques de técnicas de caja negra y testing exploratorio. Esta estrategia es seguida en el Centro de Ensayos de Software (CES). Aunque fue pensada para un proyecto de prueba independiente, puede ser seguida con éxito como parte del proyecto de desarrollo.

Como líneas de trabajo a futuro en la gestión de las pruebas funcionales, se encuentra lograr mediciones que ayuden en la estimación de las pruebas, según el tipo de producto. Para esto, se requiere tener una base de conocimiento de distintos proyectos de prueba y conocer el esfuerzo requerido para el diseño de los casos de prueba y su ejecución por tipo de funcionalidad a probar. También resulta de interés contar con una estimación del esfuerzo de las pruebas de regresión en cada ciclo.

Referencias

[1] Myers, G. *The art of software testing, 2nd edition*, John Wiley & Sons, 2004.

[2] Beizer, B. *Software testing techniques 2nd edition*, Van Nostrand Reinhold, 1990.

[3] Kaner, C., Falk, J. y Nguyen, H. *Testing Computer Software, 2nd Edition*, John Wiley & Sons, 1999 .

[4] Black, R. *Managing the Testing Process, 2nd Edition*, John Wiley & Sons, 2002.

[5] Bach, J. "Exploratory Testing Explained", *The Test Practicioner*, 2002. Disponible en http://www.satisfice.com/articles/et-article.pdf, accedido: octubre de 2007.

[6] Pérez, B., *Proceso de Testing Funcional Independiente (ProTest)*, Tesis de Maestría en Informática, PEDECIBA Informática, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006.

[7] Pérez, B., Pittier, A., Travieso, M. y Wodzislawski, M., "Testing Exploratorio en la Práctica", *VI Jornadas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC)*, Lima, Perú, pp. 43-49, 2007

[8] Esmite, I., Farias, M, Farias, N, Pérez, B. "Automatización y Gestión de las Pruebas Funcionales utilizando Herramientas Open Source". *XIII Congreso Argentino de Ciencias de la Computación (CACIC 2007)*, Corrientes, Argentina, 2007. Disponible en: http://www.cacic2007.unne.edu.ar/papers/027.pdf, accedido: Noviembre 2007.

[9] Kit E. *Software Testing In The Real World: Improving The Process*, Addison Wesley, 1995.

[10] Bach, J. "Risk-based Testing", *Software Testing and Quality Engineering Magazine* vol. 1, nº 6, noviembre-diciembre, 1999. Disponible en: http://www.stickyminds.com/getfile.asp?ot=XML&id=5009&fn=Smzr1XDD1800filelistfilename1%2Epdf, accedido: noviembre 2007.