Testing exploratorio en la práctica

Beatriz Pérez, Amparo Pittier, Mariana Travieso, Mónica Wodzislawski
Centro de Ensayos de Software,
Universidad de la República,
Instituto de Computación,
Montevideo, Uruguay
{ bperez, apittier, marianat, mwodzis }@fing.edu.uy

Resumen

El testing exploratorio se define como el aprendizaje, el diseño de casos de prueba y la ejecución de las pruebas en forma simultánea [1]. Dentro de los servicios que brinda el Centro de Ensayos de Software (CES) está la prueba funcional independiente de productos de software desarrollados por terceros. En este artículo se describe la experiencia en el CES de realizar la prueba independiente de un producto de software donde se contaba con un plazo muy corto de tiempo. Es por esto que se decidió utilizar la estrategia de testing exploratorio.

Se describe la estrategia de planificación y el abordaje de testing exploratorio utilizado, se muestran los resultados obtenidos y se concluye sobre las ventajas y desventajas de este enfoque.

1. Introducción

El término "testing exploratorio" fué introducido por Cem Kaner, se refiere a ejecutar las pruebas a medida que se piensa en ellas, sin gastar demasiado 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 una técnica de prueba en la cual quien prueba controla activamente el diseño mientras son realizadas, y utiliza la información obtenida en la exploración para diseñar nuevas y mejores pruebas [1]. En el testing exploratorio siempre se debe tomar nota de lo que se hizo y lo que sucedió [2]. Los resultados del testing exploratorio no son necesariamente diferentes de aquellos obtenidos de la prueba con diseño previo y ambos enfoques para las pruebas son compatibles [3].

El testing exploratorio puede ser aplicado en cualquier situación donde no sea obvio cuál es la próxima prueba que se debe realizar. También es adecuado cuando se requiere obtener realimentación 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 esa área.

Se presenta en este artículo la estrategia de testing exploratorio utilizada para probar un producto de software. La prueba fue realizada por el equipo del Centro de Ensayos de Software (CES), la empresa que desarrolló el producto le contrató al CES la tercerización de las pruebas del producto.

En la sección 2 se describen los conceptos y las distintas estrategias para el testing exploratorio.

En la sección 3 se describe el contexto donde se realizó la prueba, se presenta el Centro de Ensayos de Software y se describe el producto a probar.

En la sección 4 se presenta la estrategia utilizada para probar el producto, se describe la planificación del proyecto de prueba y la estrategia de testing exploratorio aplicada.

En la sección 5 se presentan los resultados obtenidos con la prueba exploratoria. Por último en la sección 6 se presentan las conclusiones.

2 Testing exploratorio

Una estrategia básica para realizar testing exploratorio es concebir un plan de ataque general, pero que permita desviarse de él por periodos cortos de tiempo e intereses diversos. Cem Kaner lo denomina el principio del “tour bus”, las personas bajan del bus y conocen los alrededores con visiones variadas. La clave es no perderse el tour entero [1].

El éxito o el fracaso del testing exploratorio está íntimamente ligado con lo que sucede en la mente del tester. Algunas habilidades deseables en los testers son la capacidad de analizar el producto y evaluar los riesgos, utilizar herramientas y tener un pensamiento crítico acerca de lo que se sabe al momento de diseñar las pruebas. También es importante que los testers tengan la capacidad de distinguir lo observado de lo inferido; ya que las inferencias podrían inducirlos a no realizar pruebas que evidencien vulnerabilidades del producto. El uso de heurísticas desempeña un rol importante en la producción de ideas variadas. Una heurística es un mecanismo mental para generar pruebas variadas y acordes a las características del producto que se está probando, por lo que es deseable que los testers las tengan presentes al momento de realizar testing exploratorio [1].

En general, el testing meramente exploratorio requiere testers con mucha experiencia. Como ventaja se encuentra que es barato y rápido, como inconveniente, que, según algunos autores, produce escasa documentación y no facilita las medidas de cubrimiento[4].

El testing exploratorio presenta una estructura externa fácil de describir. Durante un período de tiempo un tester interactúa con un producto para cumplir una misión y reportar los resultados. Una misión describe qué se probará del producto, los tipos de incidentes que se buscan y los riesgos involucrados. La misión puede ser elegida por el tester o serle asignada por el líder de testing.

Los elementos externos básicos son: tiempo, tester, producto, misión y reportes. [1]. Esta aparente simplicidad permite desplegar una amplia gama de posibilidades para la aplicación del testing exploratorio.

2.1 Estrategias de aplicación

Desde el “estilo libre”, aplicable en las primeras etapas de construcción de un producto, cuando los requerimientos, especificaciones y funcionalidades son aún ambiguos e inestables, se puede transitar hacia formas más estructuradas y documentadas del testing exploratorio.

En la selección de las distintas estrategias para su aplicación juegan un rol preponderante las necesidades de gerenciar y medir el proceso de testing exploratorio, de formalizarlo en mayor o menor grado. El resultado de aplicar el estilo libre consiste únicamente en el reporte de los incidentes detectados. Si se aplica el testing exploratorio basado en sesiones, los resultados incluyen notas escritas sobre los pasos seguidos y las observaciones realizadas, que facilitan el aprendizaje, el gerenciamiento y la acumulación de conocimiento [1]. En la literatura surgen, además del testing exploratorio basado en sesiones otras modalidades que se describen brevemente a continuación [5].

2.1 Testing exploratorio basado en sesiones

La técnica "Session Based Test Management" de James Bach [6], consiste en organizar el testing exploratorio en sesiones documentadas adecuadamente. Una sesión de testing exploratorio comprende generalmente un itinerario, que se establece a partir de la misión y eventualmente, algunas de las heurísticas a ser usadas. Su principal ventaja radica en que a pesar de su bajo costo relativo, permite elaborar reportes de avance, registrar el itinerario seguido, gestionar y medir el proceso. Es además, adaptable y flexible. Estas características son especialmente importantes cuando se está haciendo testing independiente para un cliente. Su desventaja es que depende fuertemente de las habilidades y preparación de los testers.

2.2 Testing funcional parcial

Se usa para testear funcionalidades individuales inmediatamente luego de implementadas, con el objetivo de decidir sobre su conformidad con los requerimientos y concepciones reales del diseño. Permite una rápida retroalimentación a los desarrolladores en etapas tempranas del ciclo de desarrollo.

2.3 Testing exploratorio realizado por usuarios

En muchas organizaciones, los usuarios exploran si las diferentes funcionalidades se adecuan a los escenarios reales de su trabajo. Estos usuarios tienen además del conocimiento del negocio, roles y responsabilidades variadas, que determinan naturalmente misiones específicas para el testing exploratorio que desenvuelven, no es preciso simularlas.

2.4 Testing de humo exploratorio

Se utiliza para tener una visión global y rápida sobre el nivel de calidad de una nueva versión de un producto, liberada para probar, en particular cuando las actualizaciones se producen periódica y frecuentemente. Se recorre la lista de funcionalidades básicas para detectar defectos o cambios en las funcionalidades. Por otra parte se recorre la lista de las correcciones para verificar que realmente se hayan realizado, así como las mejoras para verificar su comportamiento desde la perspectiva del usuario final.

2.5 Testing de regresión exploratorio

Cuando existen fuertes restricciones de tiempo, recursos humanos o financieros para realizar un testing de regresión exhaustivo, el testing se concentra en las correcciones y mejoras desarrolladas. Se basa fuertemente en la experiencia del tester para explorar la posible introducción de nuevos defectos o el surgimiento de efectos negativos colaterales.

3. Contexto de su aplicación

En esta sección se presenta el Centro de Ensayos de Software y el producto a probar.

3.1. El Centro de Ensayos de Software

El Centro de Ensayos de Software (CES) [7] 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 del país.

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.

3.2. Producto a probar

El producto que se probó es una aplicación web. Algunas funcionalidades de la aplicación habían sido probadas anteriormente por el CES, por lo que eran conocidas por integrantes del equipo de pruebas. La empresa de desarrollo que contrató el servicio de testing funcional del CES, requería, que en un mes, se hiciera una prueba completa de todas las funcionalidades del producto en una nueva plataforma y manejador de base de datos. Estos cambios aumentaban la probabilidad de que muchas funcionalidades tuvieran errores.

Se definió junto con el cliente que al comenzar el proyecto de prueba la empresa de desarrollo liberaría una versión del producto, la cual se probaría durante un mes. A medida que se encontraran incidentes se reportarían al equipo de desarrollo. para que pudiera hacer las correcciones correspondientes en paralelo. Al mes, el equipo de desarrollo liberaría una nueva versión con los incidentes ya corregidos. En esa segunda versión sólo se realizarían pruebas de regresión.

La única documentación del producto disponible era el manual de usuario, aún no actualizado para la versión bajo prueba.

4. Estrategia utilizada

Se presenta en esta sección la estrategia de planificación para probar el producto y el abordaje de testing exploratorio utilizado.

4.1. Estrategia de planificación

El CES cuenta con un proceso definido para realizar pruebas funcionales independientes de productos. El proceso se llama ProTest [8] y es adaptado a cada proyecto de prueba. ProTest se basa en el análisis de riesgo del producto para definir la prioridad con que se van a realizar las pruebas. Se identifican las partes del sistema que en caso de fallar tienen las consecuencias más serias y aquellas que tienen mayor frecuencia de uso, ya que si una parte del sistema es usada frecuentemente y tiene un error, el uso frecuente hace que se tengan grandes posibilidades de que la falla aparezca. [9]

Excede el alcance de este artículo explicar en detalle el proceso seguido durante este proyecto de prueba, el cual puede ser consultado en [8], pero sí se explicará la estrategia de planificación utilizada.

En la Figura 1, se muestra la planificación del proyecto de prueba, donde se definió que la primera semana se dedicaría a la planificación y organización del proyecto de prueba, el primer ciclo de prueba llevaría 3 semanas y se tendría un segundo ciclo de 2 semanas donde se realizarían las pruebas de regresión. Debido a que el tiempo para las pruebas era muy corto y el producto tiene un gran número de funcionalidades, se trabajó con un equipo de pruebas de seis personas, dirigidas por un líder de pruebas. Al comenzar las pruebas se contaba sólo con dos personas del equipo de testing que conocían el producto. Durante la semana de planificación, los testers que no conocían el producto, lo exploraron asistidos por quienes tenían experiencia en el mismo. Se decidió que las personas con conocimiento en el producto diseñaran las sesiones de testing exploratorio, contestaran dudas respecto al funcionamiento de la aplicación y validaran los incidentes encontrados durante la ejecución de las pruebas, antes de incluirlos en el sistema de seguimiento de incidentes. Durante la semana de planificación se realizó junto con el cliente un inventario de las funcionalidades a probar, este inventario se realizó a partir de los menús y del manual de usuario de la aplicación. Se catalogaron 520 funcionalidades, se realizó el análisis de riesgo, dejando 55 funcionalidades fuera del alcance. Se planificó que en el ciclo de prueba 1 se probaran mediante testing exploratorio 465 funcionalidades. En el ciclo de prueba de regresión se verificaría que los incidentes encontrados fueron solucionados en la nueva versión del producto y pueden ser cerrados.

En la última semana del proyecto, se realizó la evaluación del mismo y el informe final del proyecto de prueba.

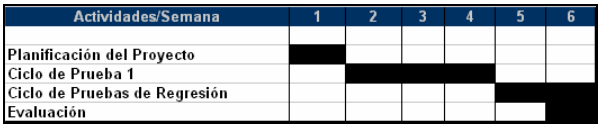

Figura 1 – Planificación del proyecto de prueba

4.2. Estrategia para Testing Exploratorio

Se optó por aplicar el testing exploratorio del producto basado en sesiones, dado lo exiguo de los plazos en relación a la cantidad de funcionalidades, y por el tipo de errores buscados.

Para definir las misiones, se estudiaron las funcionalidades de la aplicación y los ciclos funcionales. Se utilizaron dos estrategias. Por un lado, se definieron misiones en base a los principales ciclos funcionales de la aplicación y por otro, agrupando funcionalidades relacionadas. Las misiones estaban orientadas a probar que el cambio de plataforma y de manejador de base de datos no afectara el comportamiento esperado.

Las personas del equipo que conocían el producto diseñaron cinco misiones basadas en ciclos funcionales y diez misiones con funcionalidades relacionadas entre sí (por ejemplo, aquellas que pertenecían a un mismo menú o módulo), para que el resto de los integrantes del equipo llevaran a cabo sus sesiones. Luego, algunas de estas misiones fueron refinadas durante el proyecto.

Cada sesión se correspondía exactamente con una misión, y una misma misión podía ser objeto de varias sesiones.

Se definió como estrategia a seguir, que las misiones que cubrían las funcionalidades de mayor prioridad fueran asignados a más de una persona, lográndose así un cubrimiento más exhaustivo y rico.

A partir de las misiones asignadas a los integrantes del equipo de prueba, cada persona definió cada una de las sesiones que realizó. De esta forma, diseñó y ejecutó sus pruebas, registrando los resultados obtenidos en cada sesión y reportando en el sistema de seguimiento de incidentes los incidentes encontrados.

Se utilizó la herramienta Mantis [10] como sistema de seguimiento de incidentes. Dado que cuenta con una interfaz web, a medida que los incidentes eran reportados, el cliente los validaba y les asignaba la prioridad correspondiente. Esta dinámica resultó de vital importancia, ya que se logró que ambos equipos, el del CES y el de desarrollo del cliente, pudieran, en paralelo, probar y solucionar los incidentes detectados. De esta forma, la versión corregida para las pruebas de regresión, estuvo disponible casi inmediatamente, lo que permitió cumplir con los plazos establecidos.

Para cada incidente reportado se registraba la descripción, categoría, prioridad, ciclo de prueba en el cual era detectado y módulo. Mantis asigna un identificador único al incidente y registra el tester y la fecha de ingreso en forma automática.

Se definió una plantilla para registrar la información de la ejecución de las sesiones de testing exploratorio, con los siguientes datos:

- el ciclo de prueba correspondiente
- fecha y duración en minutos
- tester que realizó la ejecución
- misión de la sesión

- funcionalidades que fueron ejercitados al realizar la sesión
- razón por la que se ejecutó cada funcionalidad: por necesidad, por ser parte de la misión o por curiosidad
- datos de prueba
- observaciones: son aquellas cosas que llamaron la atención
- identificadores de los incidentes reportados en el sistema de seguimiento de incidentes.

Para poder controlar las pruebas que se estaban realizando y conocer el cubrimiento de funcionalidades que se iba obteniendo con el testing exploratorio, se mantuvo un registro de trazabilidad de las funcionalidades ya ejercitadas. Al finalizar cada jornada de trabajo, el líder de proyecto recopilaba la información de las funcionalidades ejercitadas durante las sesiones, incluyendo los incidentes encontrados, y actualizaba el documento de trazabilidad. En función de los resultados obtenidos en cada jornada, se definían las misiones para las siguientes sesiones. Esta realimentación constante fue guiando el foco de las pruebas a lo largo del proyecto.

5. Resultados obtenidos

En esta sección se exponen los resultados obtenidos en el proyecto. En la sección 5.1 se presentan las funcionalidades probadas, en la sección 5.2 se detalla cómo fueron llevadas a cabo las sesiones, y en la sección 5.3 se presentan los incidentes encontrados.

5.1. Funcionalidades

A medida que se fueron identificando las misiones, las funcionalidades complejas se descompusieron en otras más simples, obteniéndose un inventario más detallado con mayor cantidad de funcionalidades respecto al inventario inicial. Se había planificado probar, mediante testing exploratorio, 465 funcionalidades; sin embargo, en la práctica se probaron 607.

5.2. Sesiones

Las sesiones se realizaban de forma individual, cada integrante mantenía una copia impresa de la misión asignada. Antes de comenzar con la sesión, se leía la misión, y de ser necesario se aclaraban las dudas con quien la había diseñado. A continuación, se fijaba el itinerario de la sesión y se procedía a su ejecución. Si se presentaban dudas de los resultados esperados mientras se ejecutaba la sesión, se consultaba a los integrantes del equipo que tenían más conocimiento de la aplicación, el manual de usuario existente o al cliente directamente.

El tiempo registrado en cada sesión incluía el de ejecución de las pruebas y el de registro en el sistema de seguimiento de los incidentes encontrados. La duración promedio de las sesiones dependió de la persona que ejecutaba la sesión. Uno de los miembros del equipo mantuvo sesiones de menos de 1 hora de duración en promedio, mientras que otro integrante mantuvo sesiones de 3 horas de duración en promedio.

En el proyecto se definieron 20 misiones para las cuales se realizaron un total de 40 sesiones entre los 6 miembros del equipo de pruebas.

En la tabla 1 se muestra la cantidad de sesiones ejecutadas por misión. Las misiones se muestran ordenadas por prioridad, siendo 1 la más alta. Estas prioridades fueron asignadas a partir del análisis de riesgo realizado en la etapa de planificación. A partir de la tabla, puede observarse que para la misión de mayor prioridad se ejecutaron 13 sesiones mientras que, para 2 misiones de prioridad 4 se ejecutaron 2 sesiones por cada una de las misiones.

Prioridad de las misiones	Cantidad de misiones	Cantidad de sesiones por misión
1	1	13
2	1	5
3	1	3
4	2	2
5	15	1

Tabla 1 – Cantidad de sesiones por misión

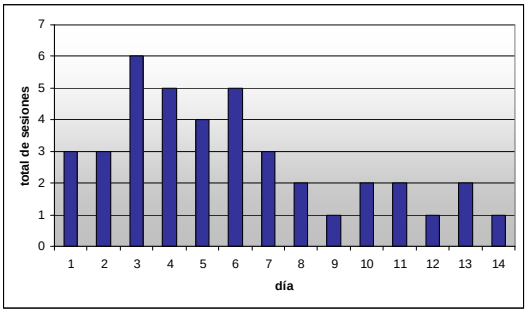

Figura 2-Total de sesiones por día

A pesar de existir una plantilla para registrar las sesiones, los documentos obtenidos diferían en contenido dependiendo del responsable de la sesión. Por ejemplo, algunos miembros del equipo describían de forma detallada sus pruebas, indicando para cada pantalla, todos los valores que le habían asignado a las entradas.

De la información registrada durante las sesiones se concluye que se invirtió un total de 100 horas para la ejecución de las sesiones. En la Figura 2 se muestra la cantidad de sesiones realizadas por día.

5.3 Incidentes

Durante el proyecto se encontraron un total de 120 incidentes. En la Figura 3 se observa que en el 25% de las 607 funcionalidades probadas se encontraron incidentes, o sea, se detectaron 154 funcionalidades con incidentes.

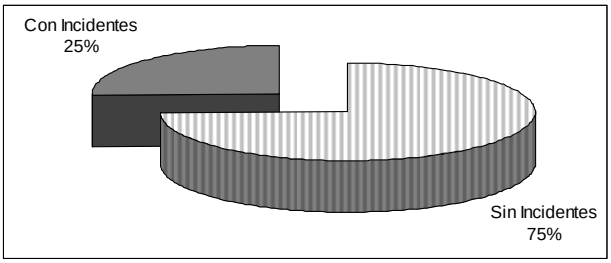

Figura 3- Cubrimiento de funcionalidades

El hecho de que la cantidad de funcionalidades con incidentes es mayor que la cantidad de incidentes encontrados se debe a que algunos incidentes fueron detectados en la ejecución de ciclos funcionales, con lo cual, un mismo incidente estaba involucrado con más de una funcionalidad.

Los incidentes fueron clasificados por prioridad: urgente, alta, normal o baja. A medida que iban siendo reportados, las prioridades asignadas eran validadas por el cliente usando el sistema de seguimiento de incidentes Mantis. En la Figura 4 se muestra los incidentes encontrados clasificados por prioridad.

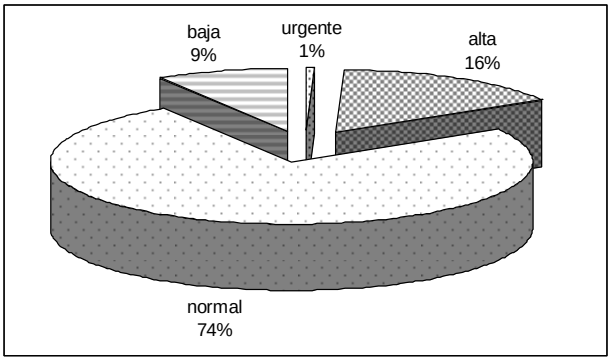

Figura 4- Incidentes por prioridad

Si bien había testers que inicialmente no estaban familiarizados con la aplicación y además se contaba con poco tiempo, se logró detectar un 16% de incidentes de prioridad alta y un 74% de prioridad normal. El cliente se mostró conforme con la calidad de los incidentes encontrados. Si bien la mayoría se debieron al cambio de plataforma, se detectaron incidentes funcionales que no estaban relacionados con dicha migración, y que no eran conocidos por el cliente. Esto agregó valor a los resultados obtenidos en el proyecto, cumpliéndose el objetivo de encontrar errores importantes en un corto período de tiempo.

6. Conclusiones

Se presentan en esta sección los aspectos positivos a destacar y los aspectos a mejorar en base a los resultados obtenidos en el proyecto presentado.

6.1 Aspectos positivos a destacar

El primer aspecto a destacar es la satisfacción del cliente con el proyecto en su conjunto, que fue considerado totalmente exitoso. El cliente consideró relevantes muchos de los incidentes encontrados y se mostró conforme con el cubrimiento de funcionalidades alcanzado.

Otro aspecto positivo es que se cumplió con la planificación del proyecto. El equipo de testing logró superar obstáculos tales como el desconocimiento de la aplicación a probar de algunos de sus miembros y la corta duración del proyecto. La primera conclusión es que hubiera sido imposible cumplir con un proyecto de estas dimensiones, crítico para el negocio y en un plazo tan exiguo, si no se hubiera aplicado testing exploratorio.

Respecto a la estrategia seguida durante las pruebas, resultó una buena práctica guiar las misiones en base a los resultados que se iban obteniendo, porque reforzó el análisis de riesgo realizado en la etapa de planificación del proyecto, poniendo el énfasis en los ciclos funcionales más críticos, con mayor cantidad de incidentes o menos probados.

Los informes de avance diarios generados y la información disponible en el Mantis, permitieron al cliente tener visibilidad del avance del proyecto en todo momento, y además tener conocimiento de los módulos y/o funcionalidades donde se concentraban los incidentes. A medida que los incidentes eran reportados los desarrolladores los iban solucionando para la versión siguiente, permitiendo comenzar con el ciclo de regresión inmediatamente después de culminar el ciclo de prueba 1. Cabe destacar que la gestión del testing exploratorio basado en sesiones permitió una comunicación fluida y permanente con el cliente.

6.2 Aspectos a mejorar

Se describen a continuación algunos de los aspectos a mejorar en la forma de trabajo.

Se plantea unificar los criterios de redacción de las sesiones, de forma análoga a lo logrado para el reporte de incidentes, que contribuyó a la buena comunicación con el equipo de desarrollo del cliente. Si bien se definió una plantilla para las sesiones, los testers la trabajaron en forma heterogénea, lo cual no facilita su lectura por parte del cliente.

Otra área de mejora es la utilización de herramientas de automatización de las sesiones para su reproducción en los ciclos de regresión. De esta forma, podría aumentarse la productividad de los testers experimentados que podrían dedicarse a nuevas funcionalidades o abordar las mismas con nuevas misiones.

La construcción de una herramienta que automatice el procesamiento de la información reportada en las sesiones es también un objetivo importante desde el punto de vista gerencial, ya que la recopilación de la información en forma manual llevó un tiempo considerable del proyecto.

Como resultado de todas las mejoras anteriores combinadas, se obtendrían métricas y mediciones más precisas, como por ejemplo, la cantidad de incidentes detectados por sesión, el tiempo relativo de preparación y ejecución de las pruebas en cada sesión.

7. Referencias

[1] J. Bach, "Exploratory Testing Explained", The Test Practicioner, 2002, http://www.satisfice.com/articles/et-article.pdf

[2] C. Kaner, J. Falk, H. Nguyen, "Testing Computer Software, 2nd Edition", ISBN: 0471358460, Editorial Wiley, 1999.

[3] J. Bach, "What is Exploratory Testing? And How it differs from Scripted Testing", StickyMinds, Enero 2001.

[4] R. Black, "Managing the Testing Process, 2nd Edition". ISBN 0-471-22398-0, Editorial Wiley, 2002.

[5] Juha Itkonen and Kristian Rautiainen, "Exploratory Testing: A Multiple Case Study", Helsinki University of Technology, Software Business and Engineering Institute, pp 84-93, IEEE, 2005.

[6] J. Bach, "Session Based Test Management", Software Testing and Quality Engineering Magazine Vol.2, No. 6, Noviembre 2000, http://www.satisfice.com/articles/sbtm.pdf.

[7] Centro de Ensayos de Software (CES), http://www.ces.com.uy, 2006.

[8] B. Pérez, "Proceso de Testing Funcional Independiente (ProTest)", Tesis de Maestría en Informática, PEDECIBA Informática, Instituto de computación, Facultad de Ingeniería, Universidad de la Republica, Uruguay, ISSN: 0797-6410 - 06-11, 2006.

[9] E. Kit, "Software Testing In The Real World: Improving The Process", ISBN 0201877562, Addison Wesley, 1995.

[10] Mantis, http://www.mantisbt.org/index.php, 2006.

Agradecimientos

El presente trabajo ha sido desarrollado en el marco del proyecto COMPETISOFT (Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria de software de Iberoamérica) del programa CYTED (Ciencia y Tecnología para el Desarrollo)