Raccis, 1(1), 15-18, 2011.

Revista Antioqueña de las
Ciencias Computacionales y la Ingeniería de Software

ISSN: 2248-7441

www.fundacioniai.org/raccis/index.htm
raccis@fundacioniai.org

Software Testing and its Relationship to the Context of the Product

Las Pruebas del Software y su Relación con el Contexto del Producto

Giordano Soares[1]
[1] Universidad de Oporto. Oporto, Portugal. G*sres(AT)fe.up.pt*

INFORMACIÓN DEL ARTÍCULO

Tipo de artículo
Reflexión

Historia del artículo
Recibido: 20-06-2011
Correcciones: 02-10-2011
Aceptado: 05-10-2011

Categories and Subject Descriptors
D.2.5 [**Software Engineering**]: Testing and Debugging – *Code inspections and walk-throughs*.

General Terms
Computer Science, Software Engineering, Software Testing, Verification, Validation.

Keywords
Test Plan, Software Errors, Domain of Product, Software Product.

Palabras clave
Plan de Pruebas, Errores del Software, Dominio del Producto, Producto Software.

ABSTRACT
Test equipment today have to deal with the growing complexity of the systems under test, while the schedules, budgets and team sizes are not scaled in the same relationship. In addition, the entire test plan cannot be automate and due to time and budget constraints, only a small fraction of all use cases can be covered in the manual tests. For this, the testers must be familiar with the context of the product under test, to design a test plan effective. This article describes the advantages that lead the fact that the testers know the domain of the software product, especially in relation to managing the complexity and problems of selection and prioritization of tests.

RESUMEN
Los equipos de prueba de hoy tienen que hacer frente a la creciente complejidad de los sistemas bajo prueba, mientras que los horarios, los presupuestos y el tamaño de los equipos no se escalan en la misma relación. Además, no se puede automatizar todo el plan de pruebas y, debido a las limitaciones en tiempo y presupuesto, sólo una pequeña fracción de todos los casos de uso puede ser cubierta en las pruebas manuales. Por esto, los probadores deben estar familiarizados con el contexto del producto bajo prueba, para diseñar un plan de pruebas eficaz. En este artículo, se describen las ventajas que conlleva el hecho de que los probadores conozcan el dominio del producto software, especialmente en lo relacionado con la gestión de la complejidad y de los problemas de selección y de priorización de las pruebas.

1. INTRODUCCIÓN

En estos días, los sistemas software se utilizan en los hogares, en las empresas, en los sistemas comerciales y para los sistemas masivos de comunicación. Sin embargo, la mayoría de personas ha tenido alguna experiencia con un producto software que no funcionó como se esperaba; por ejemplo, un error en una factura, un retraso en la espera de un proceso con tarjeta de crédito o un sitio web que no se ve correctamente. No todos los sistemas software llevan el mismo nivel de riesgo y no todos los problemas tienen el mismo impacto cuando se producen. Un riesgo es algo que no ha sucedió aún y puede que nunca suceda, es un problema potencial. La sociedad se preocupa por estos posibles problemas porque si una persona ha sufrido alguno de ellos el impacto siempre será negativo. Cuando se habla de riesgos, se debe considera la probabilidad de que el problema se produzca y cuál sería el impacto en caso de que ocurra.

A pesar de esto, en todos los contextos de los productos software no se pueden aplicar las mismas pruebas, en parte porque los distintos productos software tienen distintos requisitos, funciones y propósitos. Por ejemplo: una prueba que se diseña para un sitio web, puede no ser eficaz para una aplicación de intranet; la que se diseña para un módulo de forma de pago con tarjeta de crédito puede ser innecesariamente rigurosa si se aplica a un foro de discusión y el software crítico se prueba de forma diferente que un producto comercial. En general, cuanto mayor sea la probabilidad y el impacto de los daños causados por un error, mayor será la inversión en la realización de las pruebas.

Algunos de los problemas que se encuentran al utilizar software son bastante triviales pero otros pueden ser costosos y perjudiciales, como la pérdida de dinero, tiempo de negocios e incluso causar lesiones o muertes. Que una interfaz de usuario tenga errores tipográficos no

importa mucho y puede ser una cuestión trivial, pero podría tener un efecto significativo dependiendo de la página web y del error. Por ejemplo, si es una página con un árbol genealógico en el que aparece equivocado el nombre de la abuela materna, alguien de la familia podría molestarse, pero ese error se puede corregir fácilmente y poca gente se daría cuenta; si la página web de una empresa tiene errores ortográficos, algunos clientes potenciales podrían decidir no utilizar la empresa debido a que les parece poco profesional. Pero, si un programa no calcula bien la cantidad de plaguicida a aplicar, el efecto podría ser muy significativo: supóngase un punto decimal mal colocado de manera que la tasa de aplicación sea diez veces más grande. Este error incrementaría los riesgos, tendría impactos ambientales sobre la fauna y los suministros de agua e impactaría la salud y la seguridad para el agricultor, el jardinero, la familia, los empleados, el ganado y los animales domésticos. También se podría presentar la consiguiente pérdida de confianza y de negocios para la empresa y los posibles costos legales y las multas por causar problemas ambientales y de salud.

Todo esto justifica el hecho de que las pruebas se deben estructurar, diseñar y aplicar de acuerdo con el contexto del producto, y que los ingenieros de pruebas deben ser conocedores de ese contexto para poder aplicarlas. No se puede esperar que un probador, sin tener conocimiento del dominio o del contexto de aplicación del producto, diseñe y aplique eficientemente un plan de pruebas.

2. PRINCIPIOS DE LAS PRUEBAS

La prueba del software es una tarea extremadamente creativa e intelectualmente desafiante. El elemento creativo del diseño y la ejecución de la prueba rivaliza con cualquiera de las fases del desarrollo de software, cuando el plan de pruebas respeta los siguientes principios [1]:

2.1 *Las pruebas muestran la presencia de errores*

Probar una aplicación sólo puede revelar que existe uno o más errores en ella, sin embargo, las pruebas por sí solas no pueden probar que la aplicación está libre de ellos. Por lo tanto, es importante diseñar casos de prueba que encuentren la mayor cantidad de errores posible [2].

2.2 *La prueba exhaustiva es imposible*

A menos que la aplicación bajo prueba tenga una estructura lógica muy simple y valores de entrada limitados, no es se podrán probar todas las posibles combinaciones de datos y escenarios. Por esta razón, el riesgo y las prioridades se utilizan para concentrarse en los aspectos más importantes a probar.

2.3 *Pruebas tempranas*

Cuanto más temprano en el ciclo de vida del producto se inicien las actividades de prueba, mejor se puede utilizar el tiempo disponible [3]. Tan pronto se tengan los productos iniciales, como la elicitación y la especificación de requisitos, se debe empezar a probar; inclusive, se puede comenzar desde el momento mismo de la comprensión del problema a resolver. En la Ingeniería de Software tradicional, no es raro que las pruebas se lleven a cabo al final del ciclo de desarrollo, es decir, cuando se ha terminado el desarrollo de la solución, por lo que, si se ejecutan las pruebas tempranas, se podrán preparar pruebas para cada nivel del ciclo de vida del producto.

Otro punto importante acerca de las pruebas tempranas es que cuando los errores se encuentran al comienzo del ciclo de vida, es mucho más fácil y económico corregirlos. Es mucho más barato modificar un requisito incorrecto que tener que cambiar una funcionalidad en un sistema grande, que no funciona de acuerdo con la especificación o el diseño.

2.4 *Agrupación de errores*

Durante las pruebas, se puede observar que muchos de los errores reportados están relacionados con un pequeño número de módulos, dentro de un sistema que contiene la mayoría de los errores encontrados. Esta es la aplicación del Principio Pareto para las pruebas de software: Aproximadamente el 80% de los problemas se encuentran en el 20% de los módulos [4].

2.5 *La paradoja del pesticida*

Si se ejecuta el mismo conjunto de pruebas una y otra vez, lo más probable es que no se encuentren nuevos errores al correr los casos de prueba. Debido a que el sistema evoluciona, muchos de los errores reportados previamente han sido encontrados por casos de prueba que ya no se aplican. Cada vez que se fija un error o se agrega una nueva funcionalidad se deben ejecutar pruebas de regresión, para asegurar que las modificaciones al software no afectan ninguna otra parte del sistema. Sin embargo, los casos de prueba de regresión también tienen que cambiar para reflejar los cambios realizados en el software, con el fin de ser aplicables y, con suerte, poder encontrar posibles nuevos errores [5].

2.6 *Las pruebas son dependientes del contexto*

Las diferentes metodologías, técnicas y tipos de pruebas están relacionadas con el tipo y la naturaleza de la aplicación [6]. Por ejemplo, una aplicación de software médico necesita más pruebas que un software de juegos. Más importante aún, un software médico requiere de pruebas basadas en riesgos, cumplir con regulaciones de la industria médica y, posiblemente, de técnicas de prueba de diseño específico. De la misma manera, un *website* muy popular tiene que pasar por rigurosas pruebas de rendimiento, así como pruebas de funcionalidad, para asegurar que el rendimiento no se vea afectado por la carga en los servidores.

2.7 *La ausencia de errores es una falacia*

El hecho de que las pruebas no encuentren ningún error en el software, no quiere decir que esté listo para producción. Las pruebas aplicadas fueron diseñadas realmente para detectar la mayoría de los errores, o se concibieron para determinar si el software cumplía con los requisitos del usuario. Existen muchos otros factores que se deben considerar antes de tomar la decisión de poner el software en producción. En la Ingeniería de Software, conocer el dominio es conocer el entorno en el que el sistema operará, y cada aplicación utiliza diferentes

metodologías, técnicas y tipos de pruebas que están relacionadas con su tipo y naturaleza. Los diferentes productos software tienen diferentes requisitos, funciones y propósitos por lo que no se pueden aplicar, en todos los productos, los mismos casos y estrategias de prueba [6]. Aquí cabría anotar el mismo ejemplo del software médico de antes.

Las pruebas, como profesión, requieren que el probador sea competente, con conocimientos técnicos y con experiencia del ciclo de vida de desarrollo, dominio funcional y un uso adecuado de las tecnologías y las herramientas utilizadas para la prueba. Además, requieren que el individuo demuestre una alta capacidad de análisis lógico, de deducción, de razonamiento, de comunicación y de habilidades comunicativas.

El conocimiento del dominio del negocio es importante debido a que garantiza que el probador comprenda el punto de vista y las necesidades de los usuarios, para quienes se desarrolla el software. Un buen conocimiento del dominio del negocio asegura que los probadores sean más propensos a identificar requisitos faltantes e incompletos, o historias que podrían requerir detalles adicionales; con esto se puede identificar pruebas relevantes para requisitos/escenarios y crear casos de prueba efectivos. A menudo, la comunicación entre los usuarios finales y los desarrolladores de software es difícil, por lo que deben encontrar un lenguaje común para comunicarse; pero, desarrollar un vocabulario compartido, suficiente para comunicarse, puede tomar algún tiempo.

Cuando se prueba un producto en particular, se trabaja sobre dos tipos de ejes: el eje Y, que representa la tecnología y el eje X, que representa el dominio [7]. El dominio es común para todas las tecnologías. Los especialistas en el dominio pueden escribir los escenarios que direccionan al negocio, mientras que los especialistas en tecnología serán capaces de escribir escenarios acerca de la tecnología, con lo que su alcance es limitado.

3. VENTAJAS DE CONOCER EL DOMINIO

1. Reduce el tiempo de entrenamiento. Una persona con conocimiento del dominio puede ser productiva más rápidamente que otra que no tiene ese conocimiento, por lo que añade valor al proyecto/producto.

2. Permite un buen conocimiento del flujo funcional, los procesos de negocio y las reglas del negocio. Esto ayudará a comprender de mejor forma los requisitos del producto.

3. Se logra un buen conocimiento de las características de la interfaz de usuario. Esto ayuda a mejorar la apariencia de la interfaz, lo mismo que a descubrir más errores en las fases iniciales de su diseño.

4. Se alcanza un buen conocimiento del procesamiento *back-end*. Esto es, conocer cómo manejar el grado de eficacia/eficiencia del código/datos.

5. El conocimiento del dominio también es importante para clasificar los errores. Conocer cómo se utilizará la aplicación y la forma en que se espera que lo realice, le servirá al control de calidad para determinar si un error es trivial, importante o crítico.

6. Se adquiere una buena terminología para reportar informes en el lenguaje del negocio. Lo que resulta en una buena relación entre el equipo de trabajo.

En productos para la Banca, los Servicios Financieros o los Seguros lo más probable es que no sólo se pruebe la interfaz de usuario, la funcionalidad, la seguridad, la carga o el estrés. Por lo que se deben conocer los requisitos de los usuarios, los procedimientos de trabajo, los fondos comerciales, la exposición en la bolsa, etc. y, en consecuencia, poder probar la aplicación. Sólo entonces se puede decir que la prueba es suficiente. Aquí es donde son necesarios los especialistas de dominio. Por ejemplo, un proyecto del mercado de valores, donde se necesita conocer el concepto básico y la terminología utilizada en ese contexto. Si se conoce mejor el dominio funcional, será posible escribir y ejecutar un mayor número de casos de prueba, que puedan simular efectivamente las acciones del usuario final. Esta es una clara ventaja, y los probadores deben aplicar su conocimiento del dominio en cada paso del ciclo de vida del producto bajo prueba.

Mientras se prueba una aplicación se debe pensar como un usuario final, debido a que ese usuario puede tener un buen conocimiento del dominio en el que está trabajando. Es necesario equilibrar todas estas habilidades de manera que se abarque todos los aspectos del producto. Si el probador tiene estas habilidades, sin duda tiene la ventaja del dominio y la experiencia de la prueba y, por tanto, tendrá una mejor comprensión de los diferentes temas y podrá ejecutar, a tiempo, una mejor prueba.

4. ESTUDIO DE CASO: PRUEBA DE APLICACIONES BFSI (Banca, Servicios Financieros y Seguros) [8]

Las pruebas siempre han sido un reto en el ciclo de vida del desarrollo de un producto. La prueba es una cuestión crítica en el ciclo de vida de cualquier actividad de desarrollo de software y, en general, las pruebas de los productos financieros requieren más conocimiento del dominio. En comparación con los productos de otros dominios, los productos financieros plantean más desafíos debido a su complejidad en el diseño, el desarrollo y la prueba, el impacto en los usuarios finales y las características de seguridad. A la vez, esto significa que los ingenieros de prueba enfrentan múltiples desafíos durante las pruebas al software financiero para ofrecerle alta calidad al cliente, como se observa en la Tabla 1.

5. CONCLUSIONES

- El conocimiento del dominio juega un papel importante en la prueba de software; como dice uno de los principios de las pruebas: La prueba es conducida por el contexto [9].

- La prueba siempre se realiza de forma diferente en diferentes contextos.

Tabla1
Desafíos para los probadores de software financiero

ID	Desafío	Explicación
1	Dominio y conocimiento del sistema.	La falta de conocimiento del dominio y del sistema dará lugar a pruebas inadecuadas. Para los probadores, el conocimiento de la prueba no es suficiente para probar el producto, también deben conocer la funcionalidad completa, los escenarios de uso, los flujos de trabajo, el contexto, los estándares utilizadas, las configuraciones, las regulaciones, la interoperabilidad, las herramientas del dominio, etc.
2	Aplicación del flujo de trabajo.	Los probadores deben tener buen conocimiento del manejo y del flujo de trabajo de los casos de uso de uso del producto que ejecutan los usuarios finales en sus sitios de trabajo. Los errores surgidos por casos de uso incorrectos pueden ser rechazados, lo que conducen a la pérdida de esfuerzo en la búsqueda y el reporte de esos errores.
3	Datos de prueba y conjuntos de datos.	La falta de disponibilidad de datos de prueba y de conjuntos de datos estándar durante las pruebas dará lugar a una insuficiente cobertura de la prueba. Los ingenieros de pruebas deben tener conjuntos de datos estándar, clasificados en función de diversos parámetros y almacenados en un repositorio central compartido que esté disponible para asegurar una adecuada cobertura en los diferentes niveles.
4	Conocimiento de los estándares del dominio.	Un conocimiento insuficiente de los estándares del dominio tendrá como resultado una baja calidad de las pruebas.
5	Pruebas especializadas.	Los productos financieros se deben probar mediante pruebas específicas especializadas para cumplir con los objetivos de calidad. Los ingenieros de pruebas deben poseer experiencia, estar plenamente capacitados y poseer habilidades especiales para realizar estas pruebas.
6	Cumplimiento de procesos como ISO y CMMI.	Las unidades financieras deben cumplir ciertos estándares para certificar que son aptas para un determinado uso. Los ingenieros de pruebas deben estar capacitados en estas normas para realizar bien su trabajo al calificar el producto durante la prueba.
7	Discusión con usuarios finales o especialistas en la aplicación y el sitio.	La falta de discusiones, acerca de los casos de uso con el usuario final y de la interacción con especialistas y no especialistas en la aplicación y el sitio, darán como resultado un conocimiento insuficiente del producto y una baja calidad de las pruebas.
8	Versiones inestables.	Los desarrolladores presentan el software y el equipo de pruebas se hace responsable del producto y de la verificación. Los desarrolladores no realizan suficientes pruebas unitarias antes de presentar el producto. El código tampoco se examina suficientemente. Los ingenieros de prueba pueden proceder con la prueba de acuerdo con su plan, pero enfrentan muchos problemas que típicamente deberían haber sido capturados en las revisiones y en las pruebas unitarias.

- La experiencia en el dominio es importante en las pruebas de software, porque la persona que tiene conocimiento del dominio puede probar la aplicación mejor que otras.

6. REFERENCIAS

[1] http://istqb.org/display/ISTQB/Foundation+Syllabus, [Junio 2011].

[2] Serna, M. E. & Arango, F. I. 2011. Desafíos y estrategias prácticas de los estudios empíricos sobre las técnicas de prueba del software. Ingeniería y Competitividad, 13(1), 141-146. ISSN 0123-3033.

[3] Serna, M. E. 2011. Análisis y comparación de las propuestas recientes para diseñar casos de prueba desde los casos de uso orientados a verificar los aspectos funcionales del software. MsC. Thesis. Facultad de Minas, Universidad Nacional de Colombia Medellín.

[4] Black, R. 2002. Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing. Wiley.

[5] Black, R. 2003. Critical Testing Processes: Plan, Prepare, Perform, Perfect. Addison-Wesley.

[6] Kaner, C., Bach, J. & Pettichord, B. 2002. Lessons Learned in Software Testing. John Wiley & Sons.

[7] Black, R. 2007. Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional. Wiley.

[8] Application Software Structure Enables Nif Operations. CoRR: Software Engineering, Vol. cs.SE/0111016.

[9] Prados, M. L. 2011. El Conocimiento del Dominio como Valor Agregado al Plan de Pruebas. Lámpsakos, 9-11. ISSN 2145-4086.