GENERACIÓN DE CASOS DE PRUEBA A PARTIR DE CASOS DE USO EN LAS PRUEBAS DE SOFTWARE

Resumen / ***Abstract***

Los casos de uso son utilizados para expresar la funcionalidad requerida en un sistema. Los casos de prueba derivados directamente de casos de uso aprovechan la ventaja que constituye contar con la especificación existente de estos últimos, para garantizar un basamento sólido para las pruebas a realizar sobre el sistema. La capacidad para crear casos de prueba a partir de los casos de uso, y hacer la traza de unos a otros, es una habilidad vital para asegurar un producto de calidad.

Use cases are used to express the functionality required by a system. Use test cases derived directly from use cases take advantage of having the complete specification and documentation that exists from those last ones, to obtain a solid basement to make the necessary tests to the system. The capacity to create test cases from use cases, and to make a trace between them, is a crucial ability to ensure the quality of the software product.

Palabras clave ***/ Key words***

Prueba de software, prueba, casos de prueba, casos de uso, ingeniería de software

Software testing, test cases, use test cases, use cases, software engineering

INTRODUCCIÓN

Los casos de uso han llegado para quedarse, ya que han sido asimilados y adoptados de manera particularmente rápida por desarrolladores de todas las latitudes. No pocos especialistas han manifestado sus criterios en aras de fundamentar este fenómeno, haciendo énfasis en diversos aspectos como por ejemplo: que el modelo de casos de uso constituye el primer modelo donde se ven representados claramente los requisitos funcionales del sistema, pero sin despreciar estas apreciaciones, la reflexión, según los autores, considerada más acertada es la de Ivar Jacobson cuando apunta que "los casos de uso juegan un papel en muchos aspectos diferentes de la ingeniería de software" .[1] Lo que corrobora que una de las reglas de oro del proceso unificado de desarrollo de software (RUP) sea precisamente el ser un proceso dirigido por casos de uso: estos le dan inicio y a su vez proporcionan una guía a través de los sucesivos flujos de trabajo incluyendo el de prueba, en el que también participan de manera importante, y es alrededor de este aspecto que se desarrollará este trabajo.

En muchas organizaciones y empresas, los costos de las pruebas dentro del desarrollo de software se estiman entre el 30 y 50 % del costo total,[2] y existe la creencia por los consumidores de productos informáticos, bien fundada en no pocos casos, de que los softwares no han sido bien probados, aún cuando ya están siendo comercializados. Esta situación es a menudo provocada por el hecho de las pruebas se realizan de forma intuitiva, y no utilizando una metodología bien definida para esto, lo que concluye en la realización de un conjunto de pruebas, casi siempre en la

Odalys Jordán Enríquez, *Ingeniera Informática,Centro de Estudios de Ingeniería de Sistemas (CEIS), Instituto Superior Politécnico José Antonio Echeverría Cujae, Ciudad de La Habana, Cuba*
e-mail:ojordan@ceis.cujae.edu.cu
odyjordan@yahoo.es.com.mx

Orelvis Vázquez Ruiz, *Ingeniero Informático, CEIS, Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba*
e-mail:ovazquez@ceis.cujae.edu.cu
orelvisvr@gmail.com

Recibido: octubre del 2005
Aprobado: diciembre del 2005

etapa final del proceso de desarrollo del software, que resulta insuficiente ya que no se detecta un porcentaje elevado de las deficiencias que puede tener el producto.

Un principio básico a cumplir a la hora de iniciar el desarrollo de un producto de software debe ser el de comenzar a realizar las pruebas lo antes posible, al mismo tiempo que se va pasando por las distintas fases del ciclo de vida del software: **Inicio, elaboración** y **construcción**. Esta conclusión no es nueva, e incluso está contenida de manera global a lo largo de la especificación de RUP, donde se dedica un espacio a las pruebas en cada fase, y se especifica la labor de los ingenieros de pruebas en cada una de ellas.

A pesar de que se reconoce la importancia de las pruebas, la experiencia demuestra que resulta complicado para la Industria de software materializar la unión del inicio del desarrollo de un software con el inicio de la fase de pruebas para este, por lo que los estudiosos están buscando soluciones a esta problemática. Una de las alternativas que se propone actualmente para contribuir a iniciar un proceso de prueba temprano y a ir creando una metodología para este, es la utilización de los casos de uso para generar casos de prueba.

PRUEBAS

Para las pruebas de software, que se descomponen en varias actividades interrelacionadas entre sí, cada una con sus artefactos y trabajadores, la creación de los casos de prueba es el primer paso fundamental. A este le sigue la obtención de un conjunto de artefactos necesarios para completar el ciclo de prueba, estos son: la definición de los procedimientos de prueba a partir de los casos de prueba, que especifican cómo se van a llevar a cabo los casos de prueba; la automatización de estos procedimientos mediante la creación de componentes de prueba; la creación de planes de prueba, para definir las estrategias a seguir y los recursos necesarios para la realización de la prueba; y finalmente como resultado de este proceso la obtención de los defectos y la evaluación de la prueba realizada.[3]

Los casos de uso son clave para el proceso de prueba, no solo porque de ellos de derivan, como se ha visto, los demás artefactos, sino también porque identifican y dan a conocer las condiciones que serán implementadas en la prueba, y son necesarios para verificar que se han implementado satisfactoriamente y con calidad todos los requisitos que ha de cumplir el producto resultante.

A la hora de probar un sistema, en cada iteración se deben realizar pruebas de integración para cada construcción que se va obteniendo, con las que se mide la interacción entre los componentes de cada construcción, y al final de la iteración completa realizar pruebas de sistema, para comprobar las interacciones entre los actores y el sistema.[3] Así, se pueden identificar a lo largo del proceso dos tipos de casos de prueba:

Basados en casos de uso: Se corresponden con pruebas de sistema, donde se prueba este como caja negra, comprobando con esto el comportamiento externo visible del sistema.

Basados en las realizaciones de los casos de uso del diseño: Se corresponden con pruebas de integración donde se prueba el sistema como caja blanca, observando cómo interactúan internamente sus componentes.

A partir de lo expuesto, se puede deducir que, aunque realmente muy pocos lo hacen, los desarrolladores pueden comenzar a crear casos de prueba tan pronto sean definidos los primeros casos de uso, es decir, al comienzo del flujo de trabajo de requisitos, donde se obtiene el primer modelo de casos de uso. Se concluye entonces que mucho antes de que sea escrita la primera línea de código, ya puede darse el primer paso en el proceso de prueba que se llevará a cabo para un sistema.

CASOS DE USO

En un proyecto de desarrollo de software, los casos de uso definen los requisitos que ha de cumplir el sistema. Según RUP, un caso de uso describe totalmente una secuencia de acciones llevadas a cabo por un sistema, que proporcionan un resultado de valor a un usuario, refiriéndose este último tanto a una persona como a otros sistemas.[3]

De manera general se puede resumir que los casos de uso le "dicen" al cliente qué esperar, al programador qué programar, al documentador qué documentar y al ingeniero de prueba qué probar.

Los casos de uso pueden representarse gráficamente con los diagramas de casos de uso, y a través de una descripción textual. El diagrama representa a todos los actores que están fuera del sistema e interactúan con él, e indica a su vez cómo los casos de uso son iniciados. Cada caso de uso constituye un trozo de funcionalidad a ser implementada, y se compone de una lista de eventos iniciada por un actor, y especifica cómo este interactúa con el sistema.[4] Estos aspectos se recogen en detalle en la descripción del caso de uso, que abarca: el nombre del caso de uso, el actor que lo inicia, el propósito, un resumen, las precondiciones y poscondiciones, el flujo de eventos y los requisitos no funcionales.

Tal y como se plantea en el trabajo realizado por Jim Heumann, publicado en la revista de Rational: *The Rational Edge*, se concuerda en que la parte más importante que aporta un caso de uso para generar casos de prueba es el flujo de eventos.[2] Este se divide en dos variantes o partes: el flujo básico y los flujos alternos de eventos. La primera representa el comportamiento que se presenta con mayor frecuencia cuando se realiza el caso de uso; y la segunda abarca los casos de comportamiento opcional o excepcional con respecto a lo que normalmente ocurre. La descripción detallada de un caso de uso permite contar con una explicación completa en forma de secuencia de acciones del usuario y las correspondientes respuestas del sistema; e incluye todos los flujos de eventos posibles para el caso de uso. En el caso de los flujos alternos, se debe especificar en la secuencia de acciones, dónde comienzan en el flujo básico, y cómo terminan: si retornan al flujo básico o finalizan en sí mismos la ejecución del caso de uso.

ESCENARIOS DE CASOS DE USO

Otro elemento importante que proporcionan los casos de uso, y que hay que tener en cuenta para la generación de casos de prueba son los escenarios de casos de uso. Un escenario es una instancia de un caso de uso, o de manera similar: un flujo completo de eventos a través del caso de uso. Así, el flujo básico de un caso de uso constituye un escenario, y cada flujo alterno que se desprende del flujo básico sería otro. Por tanto, se puede decir que cada caso de uso describe varios escenarios de uso, y estos se pueden dividir en:

- Los escenarios correspondientes al flujo básico del caso de uso.
- Los escenarios que incluyen al menos un flujo alterno del caso de uso.
- Los escenarios que producen al menos una excepción.

Estos escenarios van a constituir las bases para la creación de los casos de prueba.

GENERACIÓN DE CASOS DE PRUEBA

Un caso de prueba especifica qué probar en el sistema y está formado, además del nombre y una descripción opcional, por un conjunto de entradas de prueba, de condiciones bajo las que se deben realizar las pruebas, y de resultados esperados.[3] Todo esto es realizado siempre con determinados objetivos, como puede ser la comprobación de que la parte que está siendo probada cumpla con los requerimientos para los que fue concebida y los implemente correctamente, tal como se especifica en los casos de uso. En el caso de las pruebas de sistema, que pretenden comprobar el funcionamiento de este como un todo, los objetivos pueden ser otros, en correspondencia con el tipo de prueba: pruebas de instalación, pruebas de configuración, pruebas negativas, etc. En esta temática, Ivar Jacobson dió los primeros pasos en la utilización de los casos de uso directamente en el proceso de prueba, siendo el primero en proponer la generación de los siguientes casos de prueba:[5]

1. Casos correspondientes al flujo básico del caso de uso.
2. Casos correspondientes a flujos alternos.
3. Casos que surgen de requerimientos específicos para una instancia del caso de uso.
4. Casos asociados a la prueba de características descritas en documentos asociados al caso de uso.

Más tarde, en el año 2001, Jim Heumann, trabajador de Rational SW Corporation, publica un artículo en el que describe el proceso para llevar a cabo la generación de casos de prueba directamente a partir de los casos de uso. Este proceso está basado en RUP, y tiene tres pasos:[2]

1. Para cada caso de uso, obtener el conjunto de todos sus escenarios.
2. Para cada escenario, identificar por lo menos un caso de prueba y las condiciones bajo las que será ejecutado.
3. Para cada caso de prueba, identificar los valores de entrada a utilizar en la prueba.

1. Generar los escenarios

Este paso, según los autores el más importante, se debe realizar con toda la profundidad posible, ya que al definir buenos escenarios de casos de uso se está garantizando luego, buenos casos de prueba.[6] Para la determinación de los escenarios se plantea primeramente leer cuidadosamente la descripción detallada del caso de uso, e identificar cada combinación del flujo básico con los flujos alternos, que como se vio anteriormente constituyen los escenarios. Con esta información de los escenarios se va construyendo una matriz, en la que debe quedar reflejado todo escenario que vaya a ser probado. La matriz tiene como columnas: el nombre dado al escenario, el flujo de eventos donde comienza dicho escenario (flujo básico) y el nombre del o los flujos alternos que lo componen.[2] En el caso del escenario que representa al flujo básico en su totalidad, no se le coloca un valor en la última columna ya que no incluye ningún flujo alterno.

2. Identificar los casos de prueba

Una vez que se tienen identificados todos los escenarios de casos de uso, se procede a obtener los casos de prueba. Para esto se debe analizar cada escenario y paralelamente remitirse a la descripción detallada del caso de uso correspondiente para comprobar cualquier detalle o duda que se presente. Debe haber como mínimo un caso de prueba para cada escenario, pero la generalidad es que existan más. Cuando se presenta el caso de que la descripción del caso de uso no está escrita con la profundidad requerida, y se han omitido datos importantes, para lograr obtener casos de prueba de calidad, hay que revisar dicha descripción y buscar y agregar las condiciones o elementos que faltan y que son requeridos para la ejecución de los escenarios.[2]

Para documentar los casos de prueba también se utiliza una matriz, donde cada fila se refiere a un caso de prueba. En la primera columna de esta se escribe el nombre o identificador del caso de prueba, que será utilizado en lo adelante como forma abreviada de referirlos; en la segunda columna se especifica una breve descripción del caso de prueba y el escenario que será probado. Las columnas siguientes contienen el conjunto de datos que serán necesarios para llevar a cabo las pruebas, y son llenadas en este paso con las letras: V, I y n/a, que representan el tipo de valor que contendrán (válido, no válido, no es necesario suministrarlo). Al final de estas, se añade una última columna que especifica los resultados esperados para el caso de prueba.[2]

3. Identificar los valores de entrada para la prueba

Finalmente, una vez determinados todos los casos de prueba, estos deben revisarse y validarse para asegurar su precisión, e identificar los casos de prueba redundantes u omitidos durante el proceso. Luego de esta revisión se sustituyen los identificadores generales dados a los datos en la matriz de casos de prueba (V, I, n/a), por los valores reales a utilizar, que son imprescindibles para implementar y ejecutar los procedimientos y casos de prueba respectivamente.[2]

DESVENTAJAS

A pesar de que esta técnica es utilizada por un gran número de empresas de SW, es oportuno mencionar también algunos aspectos a tener en cuenta ya que pueden verse como limitaciones de esta variante. En los casos de uso se modela la capacidad y desempeño del sistema, especificados en los requisitos adicionales del mismo, sino que se podrá encontrar en ellos los requerimientos funcionales y algunos requisitos especiales que son específicos de un caso de uso. Por tanto, los requisitos no funcionales que no queden representados en los casos de uso necesitarán ser verificados de forma independiente, fuera de los casos de prueba generados a partir de los casos de uso.

Otro elemento que no se debe pasar por alto cuando se usa este método, es la necesidad de niveles de prioridad. Dado que al final del proceso de generación de los casos de prueba, se obtiene una larga lista de escenarios y por tanto como mínimo la misma cantidad de casos de prueba, se hace necesario analizar lo obtenido y establecer un nivel de prioridad para decidir qué conjunto de casos de prueba se ejecutarán primero.

CONCLUSIONES

Derivando los casos de prueba de los casos de uso, se asegura primeramente que se ha cumplido con los requisitos de funcionalidad previstos para el sistema. Utilizando los casos de uso para generar casos de prueba, se ayuda al equipo de prueba a comenzar el trabajo desde horas tempranas en el ciclo de desarrollo, permitiéndoles identificar y corregir defectos que pueden ser muy costosos de reparar en iteraciones posteriores. Además, se asegura la entrega en tiempo de un producto confiable y a los desarrolladores les permite simplificar el proceso de prueba, incrementar la eficiencia del mismo y garantizar que en él se hayan cubierto todas las pruebas. Con este método la coordinación entre **desarrollo y prueba** está asegurada, ya que se cuenta con la organización de las pruebas obtenidas y orientadas a partir del dominio de los casos de uso del sistema.

REFERENCIAS

1. **JACOBSON, IVAR; G. BOOCH AND J. RUMBEAUGH:** *Use Cases: Yesterday, Today, and Tomorrow*, The Rational Edge, March, 2003. Disponible en: http://www.therationaledge.com/content/mar_03/f_useCases_ij.jsp
2. **HEUMANN, JIM:** *Generating Test Cases From Use Cases*, The Rational Edge, June, 2001. Disponible en: http://www.therationaledge.com/content/jun_01/m_cases_jh.html
3. **JACOBSON, IVAR; G. BOOCH Y J. RUMBEAUGH:** *El proceso unificado de desarrollo de software*. Rational SW. Corporation. Pearson Educación SA, Madrid, 2000.
4. **_____:** *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1994.
5. **JACOBSON, IVAR; M. CHRISTERSON; P. JONSSON AND G. ÖVERGAARD:** *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992.
6. **LEFFINGWELL, DEAN:** *The Role of Requirements Traceability in System Development*. The Rational Edge. September 2002, Disponible en: http://www.therationaledge.com/content/sep_02/m_requirementsTraceability_dl.jsp