XV Jornadas de Ingeniería del Software y Bases de Datos
JISBD 2006
José Riquelme - Pere Botella (Eds)

HACIA UNA PROPUESTA DE PRUEBAS TEMPRANAS DEL SISTEMA

Javier J. Gutiérrez, María J. Escalona, Arturo H. Torres, Manuel Mejías y Jesús Torres

Departamento de Lenguajes y Sistemas Informáticos
Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla
Avd. Reina Mercedes sn. 41012. España
{javierj, escalona, arturohtz, risoto, jtorres}@lsi.us.es

Palabras clave: Pruebas del sistema, generación de pruebas, pruebas tempranas

Resumen. *Una tarea vital en el desarrollo de software es probar la correcta implementación de los requisitos funcionales. Sin embargo, muchas veces la fase de prueba del sistema es demasiado corta para diseñar un buen conjunto de pruebas, ejecutarlas y analizar sus resultados. Una solución es la estrategia de pruebas tempranas, la cual consiste en obtener de manera sistemática casos de prueba en etapas tempranas del desarrollo. Este trabajo describe los puntos clave de una nueva propuesta de pruebas tempranas del sistema, la cual resuelve algunas carencias detectadas en trabajos anteriores.*

1. INTRODUCCIÓN

El incremento de la complejidad de los sistemas software incrementa a su vez la necesidad de asegurar su calidad. La fase de prueba del sistema ayuda a asegurar la calidad del software. La mayoría de las pruebas en la industria se desarrolla a nivel del sistema. Sin embargo, muchas técnicas de prueba del sistema están descritas sólo de manera informal [8]. Además, la fase de prueba del sistema suele realizarse al final del proceso de desarrollo, por lo que estas pruebas suelen realizarse de manera superficial e incompleta [8].

En este trabajo, definimos un caso de prueba como un reemplazo de un actor del sistema. El caso de prueba simula las interacciones del actor con el sistema para verificar que el sistema hace lo que se espera de él. Así, el principal artefacto para obtener pruebas del sistema son los requisitos funcionales.

Nuestra prueba temprana consiste en la generación de casos de prueba en etapas tempranas del desarrollo del software en paralelo con el desarrollo del mismo. Esta no es una idea nueva, como se verá en la sección 4. Sin embargo, dos estudios [1] y [4] (con 22 propuestas en total) muestran que existen muchas carencias en las propuestas existentes. Las propuestas existentes

no son completas. Esto significa que describen cómo obtener resultados parciales, principalmente objetivos de prueba, sin describir otros elementos importantes como datos de prueba, resultados esperados o pruebas ejecutables.
En un trabajo anterior, mostramos cómo generar casos de prueba para una aplicación web utilizando las propuestas existentes [5]. Este trabajo presenta una nueva propuesta original que intenta resolver las carencias encontradas. Nuestra nueva propuesta permite generar pruebas ejecutables a partir de casos de uso definidos en prosa y ha sido diseñada específicamente para ser aplicada en etapas tempranas, cuando el sistema aún no está construido o, incluso, diseñado. Además, nuestra propuesta se apoya en el perfil de pruebas de UML (UMLTP) siempre que sea posible [7]
La organización de este trabajo se describe a continuación. La sección 2 describe brevemente el caso práctico utilizado para ilustrar esta propuesta. La sección 3 describe la propuesta de generación temprana de pruebas del sistema a partir de los casos de uso. Finalmente, la sección 4 expone las conclusiones y los trabajos futuros.

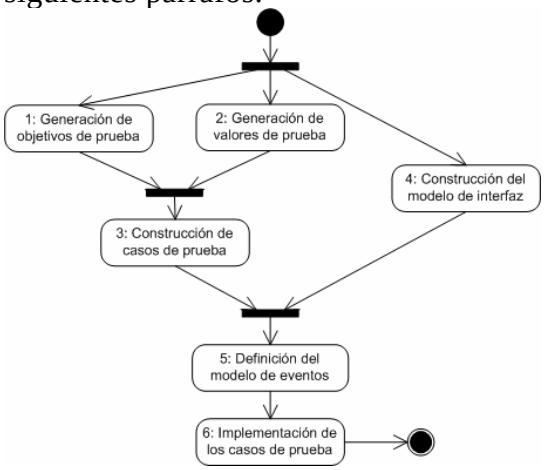
2. CASO PRÁCTICO

El sistema bajo prueba (SUT) es un editor de textos similar al Notepad de Windows. El caso de uso elegido es "Cargar documento". Este caso de uso se define en la figura 1 utilizando la plantilla para requisitos funcionales propuesta en NDT [2].

3. GENERACIÓN DE CASOS DE PRUEBA A PARTIR DE LOS CASOS DE USO

3.1. Actividades para la generación de casos de prueba

Presentamos un proceso de seis actividades para la generación de pruebas del sistema a partir de casos de uso. Dichas actividades se muestran en la figura 1 y se describen en los siguientes párrafos.



Descripción	Cargar documento
Precondición	No
Escenario principal	1 El usuario selecciona la opción de cargar un documento 2 El sistema solicita el documento a cargar 3 El usuario selecciona el fichero que contiene el documento- 4 El sistema carga el documento 5 El sistema muestra el documento.
Alternativas / Errores	3 En cualquier momento el usuario puede cancelar la operación y este caso de uso termina. 4 Si el fichero no existe o hay un error el sistema muestra un mensaje y este caso de uso termina.
Post-condiciones	No.

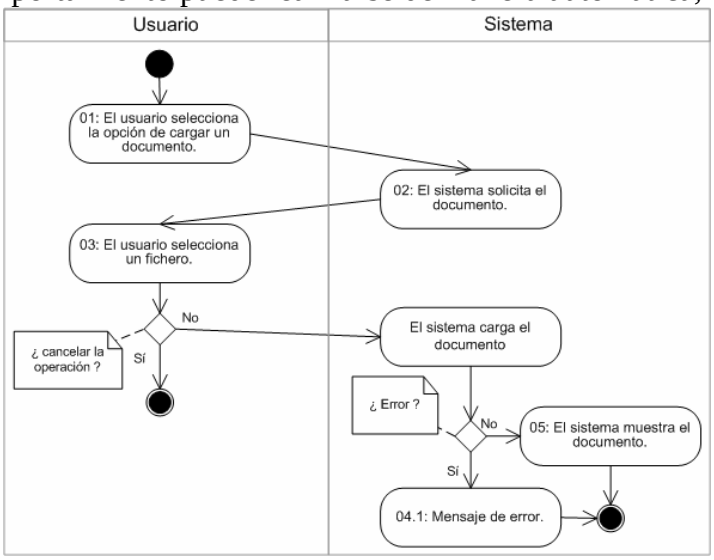
Figura 1. Actividades para la generación de pruebas tempranas.

La primera actividad consiste en construir un modelo de comportamiento y obtener

objetivos de prueba a partir de él. A continuación, en la segunda actividad, se identifican las variables del caso de uso [3] y se definen los datos de prueba. En la tercera actividad se generan casos de prueba combinando los objetivos de prueba con los valores de prueba. A continuación, en la cuarta actividad se construye el modelo de interfaz. En la quinta actividad, los objetivos de prueba se refinan y se construyen los árbitros que comprobarán si el resultado del caso de prueba es el esperado [7]. Finalmente, toda la información generada se implementa en pruebas ejecutables y test harness [3]. A continuación se definen con mayor detalle las actividades y modelos.

3.2. Generación de objetivos de prueba

El primer paso es la construcción del modelo de comportamiento. El objetivo del modelo de comportamiento es expresar la información contenida en una plantilla de caso de uso de una forma manipulable sistemáticamente. En nuestro trabajo hemos seleccionado diagramas de actividades ya que, a diferencia de los diagramas de secuencia, permiten expresar caminos alternativos fácilmente y, a diferencia de los diagramas de estados, permiten expresar la interacción entre el sistema y los actores externos identificando claramente a cada uno de los participantes. El modelo de comportamiento correspondiente al caso de uso de la figura 1 se muestra en la figura 2. La generación del modelo de comportamiento puede realizarse de manera automática, tal y como mostramos en [5].



Id	Objetivo de prueba
1	01, 02, 03, 04, 04.1
2	01, 02, 03, 04, 04.2
3	01, 02, 03

Figura 2. Modelo de comportamiento y objetivos de prueba.

En la figura 2 se muestran todos los posibles caminos obtenidos con el criterio de todos las actividades y todas las transiciones. Los objetivos de prueba, en este caso, son la cobertura de transiciones.

3.3. Generación de valores de prueba

Como se ha mencionado, en primer lugar se identifican las variables [3] El modelo de

comportamiento de la figura 2 revela dos variables. La primera, llamada OpcionUsuario, indica si el usuario selecciona la opción de cargar el fichero o cancelar la operación. La segunda, llamada Fichero, indica el fichero elegido por el usuario. Ambas variables se definen en la figura 3a y 3b.
A continuación, el dominio de las variables se divide en categorías [3], [6]. En la figura 3a y 3b se muestra una posible partición utilizando la notación propuesta por el UMLTP. Finalmente, para cada partición que se desee probar se selecciona al menos un valor de prueba. Los valores de prueba del caso práctico se muestran en el diagramas de objetos de la figura 3c.

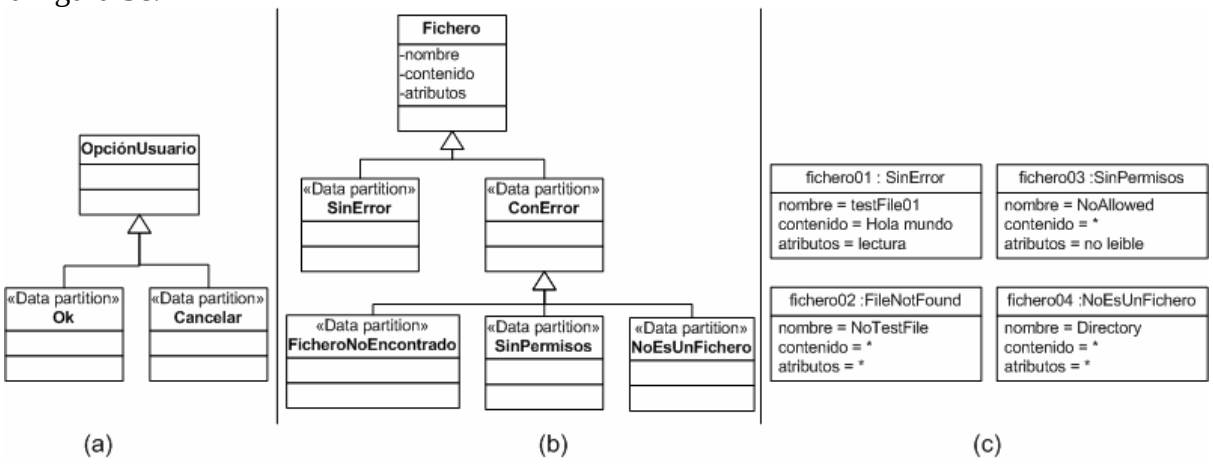


Figura 3. Datos de prueba.

3.4. Construcción de los casos de prueba

En esta actividad se combinan los objetivos y los valores de prueba. A partir de tres objetivos y dos variables se han obtenido un total de 5 casos de prueba. Todas las combinaciones se muestran en la tabla 1.

3.5. Construcción del modelo de interfaz de usuario

Al ser pruebas tempranas asumimos que el SUT aún no se ha construido y sus interfaces gráficas aún no son las definitivas (o ni si quiera se han diseñado).
Para poder refinar los objetivos de prueba se construirá una interfaz genérica con todos los elementos necesarios para que el caso de prueba pueda interactuar con el sistema. Para ello, usaremos el modelo de componentes de la figura 4a. Este modelo es fácilmente extensible.
A partir de la definición del caso de uso, se identifican las pantallas y los componentes que el sistema debe ofrecer para llevarlo a cabo. El modelo de interfaz para el caso de uso de la figura 1 se muestra en la figura 4b.

Id	Objetivo	Valor de prueba
1	1	OpcionUsuario = Ok, Fichero = fichero01
2	2	OpcionUsuario = Ok, Fichero = fichero02
3	2	OpcionUsuario = Ok, Fichero = fichero03
4	2	OpcionUsuario = Ok, Fichero = fichero04
5	3	OpcionUsuario = Cancelar, Fichero = *

Tabla 1. Combinación de objetivos y valores de prueba.

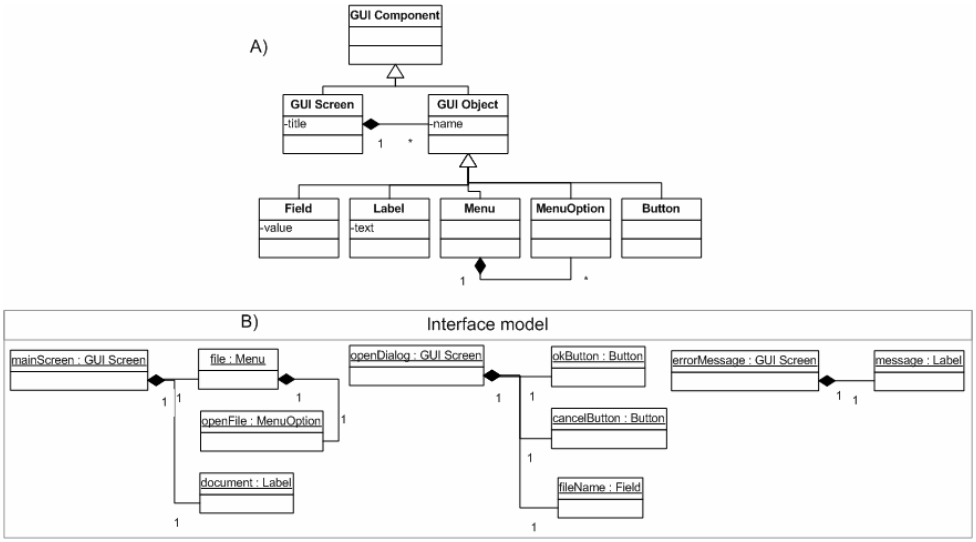


Figura 4. Modelo de componentes e interfaz del sistema.

3.6. Definición del modelo de eventos

Una vez que se dispone de la interfaz del sistema, aunque definida de una manera genérica, es posible refinar los objetivos. Para refinar las actividades realizadas por la prueba sobre el sistema proponemos un sencillo conjunto de instrucciones que se recogen en la tabla 2. A partir de las actividades realizadas por el sistema se determinan los resultados esperados y se definen los árbitros [7] que verificaran si la prueba se superó o no. Para definir los árbitros proponemos un sencillo conjunto de instrucciones que se recogen también en la tabla 2.

Instrucción	Descripción
ClickOn(component)	Representa una pulsación con el botón izquierdo sobre el componente indicado
SetField(field, value)	Asigna al campo el valor indicado.
Assert(component.attribute, value)	Verifica que el atributo del componente indicado coincide con el valor.
Screen(GUIScreen)	Verifica que la pantalla que muestra el sistema coincide con la pantalla indicada

Tabla 2. Instrucciones para expresar la interacción del usuario con el sistema.

La figura 5 muestra el modelo de secuencia y los árbitros del primer objetivo de prueba según la notación del UMLTP.

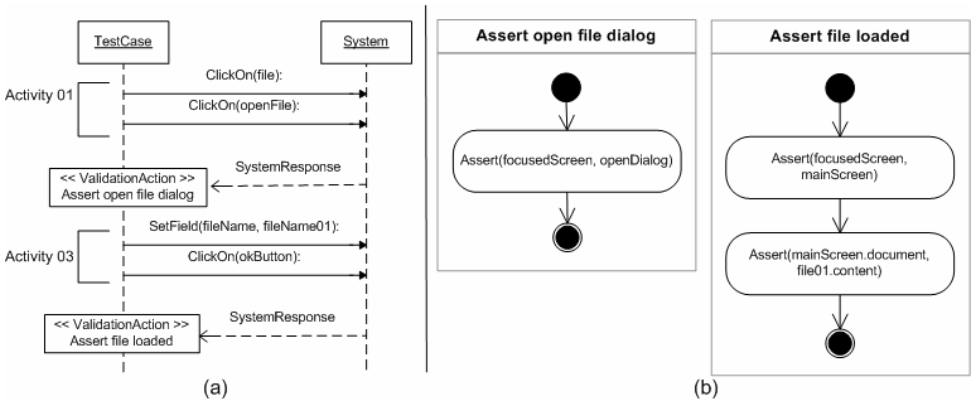


Figura 5. Caso de prueba para el primer objetivo.

El proceso de prueba temprana ha terminado. Se han generado acciones de prueba, valores de prueba y los resultados esperados. La última actividad consiste en traducir todo lo generado a pruebas ejecutables. Existen numerosas herramientas para definir pruebas. En nuestro caso práctico hemos elegido la aplicación Stylepad que acompaña al kit de desarrollo de Java y la herramienta Abbot (abbot.sourceforge.net).

4. CONCLUSIONES

Este trabajo ha mostrado un proceso para la prueba temprana del sistema a partir de los casos de uso del SUT. Aunque este proceso se ha desarrollado para probar la funcionalidad desde la perspectiva de actores humanos, puede también usarse para probar otros tipos de actores. Este proceso puede aplicarse en etapas tempranas del desarrollo. De hecho, en el caso de estudio todas las pruebas han sido diseñadas antes de elegir una implementación real. Nuestro principal trabajo es continuar refinando esta propuesta así como desarrollar herramientas que permitan su automatización. Existe una herramienta para generar diagramas de actividades a partir de casos de uso que puede descargarse en: www.lsi.us.es/~javierj/

REFERENCIAS

[1] Denger, C. Medina M. 2003. *Test Case Derived from Requirement Specifications*. Fraunhofer IESE Report.
[2] Escalona M.J. 2004. *Models and Techniques for the Specification and Analysis of Navigation in Software Systems*. Ph. European Thesis. University of Seville. Seville, Spain.
[3] Binder R. V. 2000. T*esting Object-Oriented Systems*. Addison-Wesley. USA.
[4] Gutiérrez, J.J. Escalona M.J. et-al. 2006. Generation of test cases from functional requirements. A survey. 4º Workshop on System Testing and Validation. Germany.
[5] Gutiérrez J.J. Escalona M.J. et-al. 2005. A practical approach of Web System Testing. *Advances in Information Systems Development*. pp. 659-680. Sweeden.
[6] Ostrand T. J., Balcer M. J. 1988. Category-Partition Method. *Communications of the ACM*. 676-686.
[7] Object Management Group. 2002. *The UML 2.0 Testing Profile*. www.omg.org
[8] Offutt, J. et-al. 2003. Generating Test Data from Sate-based Specifications. *Software Testing, Verification and Reliability*. 13, 25-53. USA.