Seventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2009)
"Energy and Technology for the Americas: Education, Innovation, Technology and Practice"
June 2-5, 2009, San Cristóbal, Venezuela.

Propuesta de generación de casos de prueba teniendo en cuenta indicadores de cubrimiento.

Ing. Indira Chávez Valiente
Instituto Superior Politécnico "José Antonio Echeverría", Marianao, Ciudad Habana, Cuba,
ichavez@ceis.cujae.edu.cu

MSc. Yucely López Trujillo
Instituto Superior Politécnico "José Antonio Echeverría", Marianao, Ciudad Habana, Cuba,
ylopez@ceis.cujae.edu.cu

Dra. Martha Dunia Delgado Dapena
Instituto Superior Politécnico "José Antonio Echeverría", Marianao, Ciudad Habana, Cuba,
marta@ceis.cujae.edu.cu

ABSTRACT

It is not necessary to arrive to the design stages and implementation in the cycle of life of a software to begin to think in the software tests. Diverse authors coincide in outlining that the tests should be planned from the early stages, from the moment that defines the rules of business and system requirements. This work presents a proposal of generation of procedures and test cases leaving of the specification of requirements of the software project, developed by specialists of the Center of Studies of Engineering of Systems (CEIS), of the Informatics Faculty, CUJAE. It has three phases, and particularly the last one related to the coverage of test cases designed, will be addressed with more emphasis.

Keywords: Test Cases, Functional requirements, Automatic generation of Test Cases, Coverage.

RESUMEN

No es necesario llegar a las etapas de diseño e implementación en el ciclo de vida de un software para comenzar a pensar en las pruebas a realizar sobre el mismo. Diversos autores coinciden en plantear que las pruebas deben planificarse desde las etapas tempranas, desde el mismo momento en que se definen las reglas del negocio y los requisitos del sistema. Este trabajo presenta una propuesta de generación de procedimientos y casos de prueba partiendo de la especificación de requisitos del proyecto de software, desarrollada por especialistas del Centro de Estudios de Ingeniería de Sistemas (CEIS), de la Facultad de Informática de la CUJAE. La misma cuenta con tres fases, y en particular la última de ellas, relacionada al cubrimiento de los casos de prueba diseñados, se abordará con mayor énfasis.

Palabras Clave: Casos de Prueba, Requisitos Funcionales, Generación Automática de Casos de Prueba, Cobertura.

1. INTRODUCCIÓN

El desarrollo de un sistema informático depende de múltiples factores variables a cada equipo de trabajo, tal es el caso de la tecnología a utilizar, las dimensiones y el sector sobre el que repercutirá posteriormente, el cual a su vez puede ser de los más o menos críticos en la sociedad. Sin embargo, las personas son un elemento común en todos los entornos de desarrollo, por lo tanto también serán comunes la inserción de errores, los choques de criterios y los cambios de ideas. Se vuelve necesario entonces probar el software en múltiples ocasiones. Myers, define el proceso de prueba del software como la búsqueda de fallos mediante la ejecución del sistema o de partes del mismo en un entorno y con valores de entrada bien definidos (Myers, 2004).

Desde las primeras etapas en el ciclo de vida de un software se pueden comenzar a planificar las pruebas a realizar, incluso antes de llegar a la etapa de codificación, con esto se evita insertar errores en la aplicación, los cuales pueden volverse muy difíciles de encontrar en fases posteriores. A lo anterior se añade que la mayor parte de los defectos se introducen en las etapas tempranas del proceso de desarrollo y de una manera irrefutable se ha comprobado que el costo de su corrección aumenta a medida que el error permanece no detectado (McEwen, 2004) (Usaola, 2006).

A pesar de constituir una buena práctica el hecho de definir el plan de pruebas desde las primeras etapas, muy pocos se acogen a esta idea y comienzan a pensar en las pruebas solo después de tener el código frente a ellos. Y es que desde el primer momento en que se definen las reglas del negocio y los requisitos del sistema, surgen un conjunto de situaciones inviolables las cuales deben ejecutarse satisfactoriamente surgiendo entonces las primeras propuestas de pruebas a ejecutar (Sanz, 2007).

Un **caso de prueba** especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse (Jacobson, 2004). Sin embargo este puede estar orientado a encontrar defectos en la interfaz de usuario, en la ejecución de estructuras de datos, o un determinado requisito funcional. En este sentido, un producto de ingeniería puede probarse de dos formas según (Presuman, 2005):

- Conociendo la función específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa y al mismo tiempo, buscando errores en cada función.
- Conociendo el funcionamiento del producto, se pueden desarrollar pruebas que aseguren que todas las piezas encajan, o sea que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han probado de forma adecuada.

El primer tipo de prueba es conocido como prueba de caja negra o de comportamiento, el cual se centra en los requisitos funcionales del software. Mediante las técnicas de prueba de caja negra se intentan encontrar errores en funciones incorrectas o ausentes, errores de interfaz, errores de rendimiento, inicialización y terminación. El segundo tipo es conocido como 'prueba de caja blanca', está más orientado a operaciones internas, es decir, estrechamente vinculado con la codificación. A través de las diferentes técnicas de prueba de caja blanca se garantiza que se ejecuten todos los bucles, se ejerciten todas las decisiones lógicas y las estructuras internas de datos.

La propuesta que se presenta en este artículo toma en cuenta solamente las pruebas a los requisitos del software (funcionales o de caja negra), cuyo objetivo fundamental es validar si el comportamiento observado del software cumple o no con las especificaciones.

Por otra parte están los **procedimientos de prueba**, definidos en (Jacobson, 2004) donde se dice que estos especifican como realizar varios casos de prueba o parte de estos. Por ejemplo se puede diseñar un procedimiento de prueba para insertar un nuevo estudiante en un sistema para la docencia, para ello se elaboran todos los pasos necesarios de forma consecutiva para dar alta a un estudiante en la aplicación. Sin embargo se pueden insertar estudiantes masculinos, femeninos, de mayor o menor edad por lo que al procedimiento de prueba de insertar estudiantes será necesario asociarle un grupo de casos de prueba responsables de las diversas entradas del usuario. Dicho de otra forma, los casos de prueba no están presentes de forma independiente sino que se encuentran asociados como mínimo a un procedimiento de prueba.

La mayoría de los equipos de desarrollo de software cuentan con un tiempo limitado para la creación de pruebas minuciosas y bien planificadas, lo que trae consigo un atropello de pruebas en su mayoría funcionales y sin una planificación previa. Lo anterior es equivalente a que se hagan pruebas redundantes, que haya caminos que no sean probados y que las entradas a los casos de prueba no sean adecuadamente escogidas, por lo que a una peor planificación de las pruebas a realizar, son mayores las probabilidades de dejar de analizar casos de prueba fundamentales para asegurar que el software se entregue con un nivel de calidad aceptable.

El término cobertura está más asociado a las pruebas de caja blanca donde se puede encontrar la cobertura de sentencias, de decisiones, condiciones, el cubrimiento de bucles entre otros (Usaola, 2006). Aun así también puede encontrarse relacionado a las pruebas de caja negra mediante la cobertura de los valores de entrada y la cobertura de los caminos posibles dentro de las funcionalidades del sistema.

En el presente trabajo se realiza un análisis de propuestas existentes para la generación de pruebas a partir de la especificación de requisitos funcionales, donde se han identificado los elementos comunes presentes. Por último, se presenta una propuesta de generación de procedimientos y casos de prueba partiendo de la especificación de requisitos del proyecto de software, la cual reutiliza información almacenada en un repositorio de datos. Dicha propuesta consta de tres fases, y se abordará con mayor énfasis la última de ellas, relacionada precisamente con la reducción del conjunto de casos de prueba, de forma tal que se mantenga el nivel de cobertura deseado.

2. ANÁLISIS DE PROPUESTAS EXISTENTES PARA LA GENERACIÓN DE CASOS DE PRUEBA A PARTIR DE LA ESPECIFICACION DE REQUISITOS FUNCIONALES

Dentro del campo específico de la generación automática de casos de prueba, los autores, a partir de la bibliografía consultada, han identificado varios elementos que deben tenerse en cuenta para la formulación de cualquier propuesta que se haga en este sentido. Estos elementos, a su vez, sirven de base para guiar y organizar el estudio acerca de propuestas existentes. Los elementos identificados son los siguientes:

1. **Información Base:** En sentido general las técnicas de diseño de casos de prueba existentes se basan en tomar las especificaciones como referencia de comportamiento de la aplicación. Lógicamente es importante que la especificación haya sido validada con el mayor cuidado mediante revisiones, para detectar posibles problemas. Varias propuestas parten de la especificación de casos de uso en plantillas que siguen una estructura determinada, otras de la especificación en lenguaje natural y pueden encontrarse propuestas que reciban las especificaciones en un lenguaje formal definido previamente (Gutiérrez, 2005) (Gutiérrez et al., 2005) (Méndez et al., 2007).

2. **Formato de descripción de secuencia de pasos principales o base y de pasos alternativos:** Importante tener en cuenta que de cualquier especificación que se parta esta debe describir una secuencia de pasos principales o base y una serie de pasos alternativos. Existen técnicas de análisis del lenguaje natural para la transformación de esta especificación a un grafo (Gutiérrez, 2005) (Gutiérrez et al., 2005) (Fernández et al., 2003) (Fernández et al., 2004).

3. **Algoritmo para obtener los caminos de ejecución:** Para obtener los caminos de ejecución posible, aun sin precisar los valores de prueba (sería obtener procedimientos de prueba, concepto definido previamente), existen varias métodos y algoritmos que coinciden en partir de la modelación del comportamiento mediante diagramas de estado, de actividad (caso particular de un diagrama de estado), mediante grafos (Fernández et al., 2003).

4. **Algoritmo para asignar valores a las variables que constituyen cualquier elemento que pueda variar entre dos instancias de un caso de uso:** Para la asignación de valores en el proceso de diseño de casos de pruebas existen diferentes técnicas descritas ampliamente en (Myers, 2004) (Hutcheson, 2003) (Grindal et al., 2004) (Copeland, 2004).

5. **Criterios de cobertura:** Resulta imprescindible que el diseño de pruebas se apoye en la selección de algunas entradas o situaciones que ejercitar (dentro de todo el universo posible), siempre en función de que sean representativas en su comportamiento de otros casos similares. Los criterios de selección (denominados criterios de cobertura) suelen pretender la cobertura de un porcentaje o una representación del universo posible de

elementos estructurales o de situaciones funcionales. De hecho, algunos autores hablan más de selección de casos de pruebas que de diseño de los mismos (Fernández et al., 2003).

Actualmente existen varias propuestas para la obtención de casos de prueba para la realización de pruebas al sistema a partir de los requisitos del sistema, como son: SCENT (Roser and Glinz, 2003) (Cavarra and Davies, 2004), AGEDIS (Cavarra and Davies, 2004) (Hartman, 2004), Generación de Casos de Pruebas a partir Casos de Uso (en inglés Generating Test Cases From Use Cases) (Heumann, 2003), Generación de Casos de Pruebas Estadísticos basados en UML (en inglés UML-Based Statistical Test Case Generation) y Análisis de los Caminos del Caso de Uso (en inglés Use Case Path Analisys) (Gutiérrez, 2005) .A continuación se realiza una comparación de las propuestas anteriores, teniendo en cuenta algunos factores expuestos en (Gutiérrez et al., 2005):

Tabla 1: Comparación entre procesos de reducción.

Factores / Procesos	Scent	Agedis	Casos de Prueba a partir de Casos de Uso	Análisis de Caminos en Casos de Uso
Nueva Notación	Sí	Sí	No	Sí
Automatización Completa	No	Sí	No	No
Casos Prácticos	Sí	Sí	No	Sí
Uso de Estándares	Sí	Sí	Sí	Sí
Dificultad de Implantación	Media	Alta	Baja	Baja

3. PROPUESTA DE GENERACIÓN DE PROCEDIMIENTOS Y CASOS DE PRUEBA A PARTIR DE LA ESPECIFICACION DE REQUISITOS DE UN PROYECTO DE SOFTWARE

Según (Heumann, 2004) los requisitos deben ser nombrados utilizando un único verbo y un sustantivo sobre el cual se realiza la acción. En esto se basa justamente el criterio enunciado en (Delgado, 2006) por el cual se definen un conjunto de patrones de requisitos que describen la interacción entre actores y sistema. Este trabajo ha sido punto de partida para la definición de algunos de los elementos que serán expuestos en esta sección.

La propuesta concreta del presente trabajo consiste en reutilizar en el desarrollo de un proyecto un conjunto de procedimientos y casos de prueba, de forma tal que ante la necesidad de elaborar nuevos casos de prueba estos puedan ser generados partiendo únicamente de la especificación de requisitos del proyecto de software en cuestión.

La propuesta tiene tres grandes fases: Una primera en la que se traduce la especificación de requisitos del software, escrito en un lenguaje no formal, a un listado de requisitos funcionales reducido a un verbo y un sustantivo que caracterizan la funcionalidad en cuestión (figura 1). Para realizar este proceso se utiliza la herramienta FreeLing, a la que se le suministra el requisito para obtener el verbo y el sustantivo que lo caracteriza.

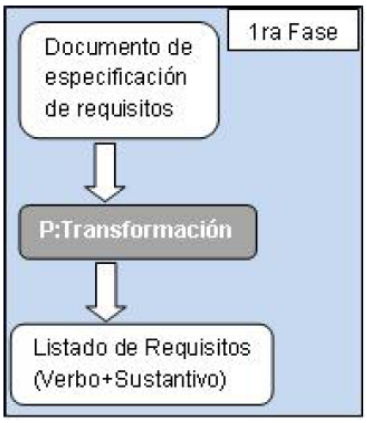

Figura 1: Fase 1 de la Propuesta

En la segunda fase se generan un conjunto de casos y procedimientos de prueba que contribuyen a probar los requisitos funcionales, tal y como se muestra en la figura 2.

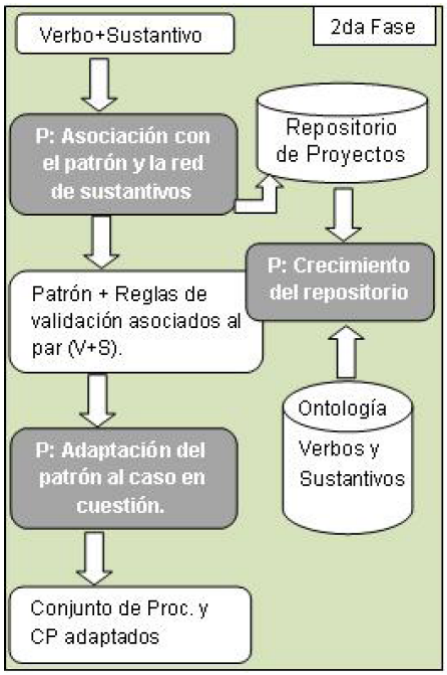

Figura 2: Fase 2 de la Propuesta

Para hacer posible la generación de los procedimientos de prueba se han diseñado un conjunto de patrones típicos de procedimientos de prueba, asociados al verbo que está contenido en el requisito funcional, partiendo de que requisitos similares deben ser probados de forma similar. El punto de partida será la asociación de un requisito con un patrón de procedimiento de prueba y la red de sustantivos correspondientes de la fase 2, lo constituye la salida del FreeLing.

Este último identifica, para cada requisito, un único verbo y sustantivo que lo caracterizan. Como parte de este proceso se llevan a cabo las siguientes actividades:

1. Asociar cada requisito con una entrada en la red de patrones típicos de procedimientos de prueba.

2. Asociar cada requisito con una entrada en la red de sustantivos.

Como resultado de llevar a cabo estas actividades se obtienen las salidas de este proceso que se corresponden con:

- Entrada en la red de Patrones típicos de procedimientos: Caminos posibles que describen el funcionamiento propio del requisito y que están contenidas en el patrón de procedimiento asociado con el verbo particular del requisito.
- Entrada en la red de sustantivos: Reglas de validación asociadas a un sustantivo que estarán almacenadas en la red de sustantivos.

Otro de los elementos que forma parte de la propuesta es un repositorio de proyectos, en el que están contenidos los elementos anteriores. Se ha definido una red semántica para almacenar la información de los patrones de procedimientos de prueba y otra para almacenar entidades con la información correspondiente a los sustantivos.

Cada entidad tiene los atributos de un sustantivo y la regla de validación asociada a cada atributo. La regla de validación contiene el conjunto de valores permitidos en el caso de ser un atributo simple, pero en el caso en que se referencia a otra entidad puede contener una condición.

Es importante aclarar que cada entidad puede ser asociada a un conjunto de sustantivos que representan objetos de la vida real que tienen un comportamiento similar y que por tanto a la luz de la generación de los procedimientos y casos de prueba pueden tener reglas de validación similares.

Las entidades tienen la siguiente estructura:

- Nombre de los objetos que representa (lista de sustantivos sinónimos).
- Nombres de los atributos (lista de atributos)
- Reglas de validación (lista de reglas de validación asociada con cada uno de los atributos)

Por último una tercera fase, en la que se reduce el conjunto de casos y procedimientos de prueba propuestos a un conjunto con determinado nivel de cobertura (figura 3).

Figura 3: Fase 3 de la Propuesta

Los procesos descritos hasta el momento son ejecutados en caso de que existan en el repositorio de proyectos las entradas identificadas por verbos y sustantivos semejantes a los del requisito que se analiza. En caso de ser un nuevo requisito el repositorio debe enriquecerse, y para este caso se ha considerado el proceso de *Crecimiento del*

Repositorio. Para llevarlo a cabo se plantea la utilización de una ontología de verbos y sustantivos que permita enriquecer semánticamente la red presente en el repositorio.

La primera y segunda fase son temas de otros trabajos, en particular en la sección siguiente profundiza en algunos elementos de los procesos contenidos en la tercera fase de la propuesta.

DESCRIPCIÓN DE LA TERCERA FASE DE LAPROPUESTA

A raíz de toda la investigación realizada sobre el tema de la cobertura en las pruebas, y los problemas que trae consigo un bajo cubrimiento en las funcionalidades del software, se ha decidido comenzar a trabajar en la definición de un conjunto de pasos los cuales incorporen los factores que influyen en la selección de un buen plan de pruebas. Estos factores o indicadores pueden estar relacionados a través de una función que modele la cobertura de un "test-suite" específico, aún por definir, para lo cual incluso se podrían utilizar técnicas de Inteligencia Artificial.

PROPUESTA DE INDICADORES PARA LA COBERTURA DE LAS PRUEBAS

En el Centro de Estudios de Ingeniería de Sistemas (CEIS) de la Facultad de Informática del Instituto Superior Politécnico José Antonio Echeverría (CUJAE), el grupo de trabajo de Calidad ha decidido fragmentar esta fase en dos etapas fundamentales, una primera en la que se seleccionen los indicadores mencionados y una segunda etapa en la que se defina y valide una función que englobe estos factores.

Esta segunda etapa será realizada sobre la nueva versión del sistema GREHU (Sistema Automatizado para la Gestión de Recursos Humanos) que se desarrolla en el Grupo de Informática Empresarial del CEIS. Cada equipo de prueba ejecutará un test-suite donde el conjunto de indicadores ha sido priorizado de manera diferente, para posteriormente proceder al ajuste de estos indicadores. Como parte de la validación se utilizarán un conjunto de métricas como las propuestas en (Delgado, 2006), entre las que se encuentran la efectividad en la eliminación de defectos y la cantidad de defectos detectados por casos de prueba.

En estos momentos la investigación se encuentra en la primera etapa de esta fase, se han seleccionado un conjunto de indicadores propuestos por los diferentes autores, que serán sometidos a criterio de expertos, utilizando el método Delphi. Estos indicadores parten de un conjunto de recomendaciones de diferentes autores para priorizar un caso de prueba por sobre otros, en el momento de seleccionar los que conformarán el plan de pruebas, de forma tal que se obtenga un determinado nivel de cobertura.

En una primera ronda se les suministró a los expertos un listado preliminar de indicadores, para que los ordenen por niveles de importancia. De esta ronda se obtuvieron los siguientes indicadores:

I1: Nivel de riesgo de cada uno de los requisitos (Bajo, Medio o Alto) asociados con el caso de prueba.

I2: La frecuencia de ejecución de un requisito determinado, entendiendo por esta frecuencia la cantidad de veces que este requisito se ejecutará dentro de la aplicación.

I3: Representatividad de valores de entrada a casos de prueba, teniendo en cuenta atributos concretos de la aplicación. Esto se basa en la idea de qué se quiere cubrir.

I4: El grado de relevancia que puede tener para la aplicación la ejecución de un camino específico dentro del requisito, para lo cual se puede tener en cuenta una escala que ayude a ponderar la importancia del mismo. Pensando, por ejemplo, que el curso normal de eventos de un requisito debe tener mayor significación que un curso alterno.

I5: El factor representatividad es decir, cuán re-presentada está una funcionalidad, o parte de ella (camino básico o cursos alternos) en el conjunto de caminos mínimos que garantizan el mayor nivel de cobertura.

Teniendo identificados los factores relacionados con la medición de la cobertura, será necesario aplicar una segunda ronda para conocer la importancia de cada uno de estos en la selección de los casos de prueba que

conforman un test suite, y alcanzar una cobertura determinada. Podría complementarse el análisis con la asignación de un peso a cada uno de los factores encontrados.

Posteriormente se compararán los resultados obtenidos en las pruebas de los conjuntos de casos de prueba seleccionados para de esta forma ajustar la función de cobertura propuesta y si fuese necesario someterlo nuevamente a criterio de expertos.

En la Tabla 2 aparecen los indicadores propuestos hasta el momento, relacionados con cada uno de los procesos y metodologías vistas, analizando la presencia de estos o no en las diferentes vías de obtención de casos de prueba incluidas en la investigación.

Tabla 2: Relación entre procesos definidos e indicadores propuestos

Factores / Procesos	Scent	Agedis	Casos de Prueba a partir de Casos de Uso	Análisis de Caminos en Casos de Uso
I1				X
I2				X
I3	X	X	X	
I4	X		X	X
I5	X			

Como se muestra en la tabla anterior, el proceso que incorpora mayor número de indicadores es Scent, a través de los 16 pasos definidos en su metodología, y es el único que tiene en cuenta el indicador de la representatividad, mediante la creación de escenarios abstractos que incorporen los elementos comunes de un grupo de escenarios.

Agedis es la propuesta que menos indicadores incluye. En la bibliografía consultada acerca de este proceso se hace énfasis en las herramientas que incorpora y su relación entre ellas, sin embargo los algoritmos y técnicas a tener en cuenta para obtener un cubrimiento adecuado son poco abordados, según (Craggs et al., 2003) estos se concentran en probar cada una de las reglas con combinaciones de valores de entrada.

Por otra parte, no hay un indicador que aparezca en todos los procesos, además, el indicador que más está reflejado en los procesos es el relativo a los valores de entrada seleccionados para los casos de prueba, lo cual es lógico debido a que es el relacionado a las técnicas ya tradicionales de las clases de equivalencia y los valores límites.

REFERENCIAS

Blanco, R., E. Díaz, and J. Tuya, "Generación automática de casos de prueba mediante búsqueda dispersa". Revista Española de Innovación, Calidad e Ingeniería del Software, 2006. Vol.2, No. 1.

Cavarra, A. and J. Davies, “Modelling Language Specification”. AGEDIS Consortium Internal Report, Disponible en: http://www.agedis.de, 2004.

Copeland, L., "A Practitioner's Guide to Software Test Design". 2004

Craggs, Ian y otros. (2003). AGEDIS Case Studies: Model Based Testing in Industry.

Delgado, M.D,D., "Un modelo para la gestión de Revisiones en proyectos de software utilizando Razonamiento Basado en Casos". Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas., 2006. Facultad de Informática. CUJAE

Fernández, L., P.J. Lara, and C. Gutiérrez, "Generación de casos de prueba a partir de especificaciones UML". Universidad Europea de Madrid, 2003.

Fernández, L., P.J. Lara, and J.J. Cuadrado- Gallego, "Mejora de la calidad en desarrollos orientados a objetos utilizando especificaciones UML para la obtención y precedencia de casos de prueba". Revista de Procesos y Métricas de las Tecnologías de la Información (RPM). Asociación Española de Sistemas Informáticos (AEMES), 2004. VOL. 1, Nº 3: p. 11-20.

Grindal, M., J.O., Sten F. A, "Combination Testing Strategies: A Survey". GMU Technical Report ISE-TR-04-05, 2004.

Gutiérrez., J.J., "Generación de pruebas de sistema a partir de la especificación funcional". Departamento de Lenguajes y Sistemas Informáticos. Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla, 2005.

Gutiérrez, J.J., et al., "Estudio comparativo de propuestas para la generación de casos de prueba a partir de requisitos funcionales". Departamento de Lenguajes y Sistemas Informáticos. Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla, 2005.

Hartman, A., AGEDIS Final Project Report AGEDIS Consortium Internal, 2004

Heumann, J., “Generating Test Cases from Use Cases”. Journal of Software, Testing Professionals, 2003.

Heumann, J., "Writing good requirements is a lot like writing good code". IBM Corporation 2004, Julio 2004.

Hutcheson, M.L., "Software Testing Fundamentals: Methods and Metrics". 2003

Jacobson, B., Rumbaugh. , El Proceso Unificado de desarrollo de Software, 2004. Volumen 1,Editorial Félix Varela: p. Capítulo 11 pag 281-304.

McEwen, S., “Requirements: An introduction”. IBM Rational, IBM Corporation, Abril 2004.

Méndez, E., M. Pérez, and L.E. Mendoza, "Aplicación de un Método para Especificar Casos de Prueba de Software en la Administración Pública". Actas de Talleres de Ingeniería del Software y Bases de Datos, Vol. 1, No. 4, 2007.

Myers, G.J., "The Art of Software Testing". Second Edition. 2004.

Pressman, R., Ingeniería del Software:un enfoque práctico. Editorial Félix Varela, 5ta edición, 2005. Volumen 1: p. Capítulo 17 "Técnicas de prueba del software".

Roser, J. and M. Glinz, “Scent: A Method Employing Scenarios to Systematically Derive Test Cases for System Test”. Technical Report 2000/03, Institut für Informatik, Universität Zürich, 2003.

Sanz, L.F., Tutorial: pruebas funcionales y trabajo en equipo. Universidad Europea de Madrid, Grupo de calidad de software de ATI., 2007.

Usaola, M.P., Curso de doctorado sobre Proceso software y gestión del conocimiento. Pruebas del Software. Universidad de Castilla-La Mancha. Departamento de Tecnologías y Sistemas de Información, 2006.

Authorization and Disclaimer