Propuesta de pruebas funcionales utilizando el Razonamiento Basado en Casos

ABSTRACT

The software companies must assess the quality of their applications and between the main types of tests are to develop function tests, which aim to ensure that the software meets the requirements, a task that is gaining increasing importance . A definite test strategy is not sufficient for the detection of most software defects. The design and implementation of software testing generally have limited time in project planning, resulting in the execution of tests without the required quality due to accumulation of test cases is large in many cases according to the complexity and size of the project. All this results in the need to use a tool that allows from historical behavior can infer the future behavior of the tests according to their characteristics. Case-Based Reasoning (CBR) allows adapt and validate the solutions on previous experience from the implementation of functional tests. This research presents a proposal to run tests using the RBC, which provides greater efficiency and quality in the results obtained at the end of the tests.

Keywords: quality, functional tests, cases, infer

RESUMEN

Las empresas productoras de software deben evaluar la calidad de sus aplicaciones y entre los principales tipos de pruebas a desarrollar se encuentran las pruebas de función, que tienen como objetivo comprobar que el software cumpla con los requisitos establecidos, tarea que va cobrando cada día mayor importancia. Una estrategia de pruebas definida, no es suficiente para la detección oportuna de la mayoría de los defectos en el software. El diseño y ejecución de las pruebas de software generalmente cuentan con tiempo limitado dentro de la planificación del proyecto, lo que ocasiona la ejecución de pruebas sin la calidad requerida debido a la acumulación de casos de pruebas que en muchos casos es grande de acuerdo a la complejidad y tamaño del proyecto. Todo esto trae como consecuencia la necesidad del uso de una herramienta que permita que a partir del comportamiento histórico se pueda inferir el comportamiento futuro de las pruebas de acuerdo a sus características. El Razonamiento Basado en Casos (RBC) permite recuperar, adaptar y validar las soluciones encontradas en experiencias previas derivadas de la ejecución de pruebas funcionales. La presente investigación presenta una propuesta para ejecutar pruebas utilizando el RBC, la cual aporta mayor eficiencia y calidad en los resultados obtenidos al concluir las pruebas.

Palabras claves: calidad, pruebas funcionales, casos, inferir

1. INTRODUCCION

La calidad del software se define como la "concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente" (Pressman, 2005). Se relaciona con conceptos como funcionalidad, usabilidad, estabilidad, escalabilidad, eficiencia, mantenimiento y

seguridad y en la actualidad, es crucial verificar estos y otros factores claves para el éxito de un sistema que cumpla las expectativas de los clientes. Para asegurar un determinado nivel de calidad se deben efectuar pruebas que permitan comprobar el grado de cumplimiento de los requisitos del sistema, las cuales deben integrarse en las diferentes fases del ciclo de vida dentro del desarrollo del software, lo cual es un factor clave para favorecer la calidad de todos los productos. Entre los principales tipos de prueba que se pueden ejecutar se encuentran las pruebas de función.

La funcionalidad de un sistema se define como: "La capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas cuando el software se usa bajo las condiciones especificadas". (NC ISO/IEC, 2003)

La prueba de función está centrada en validar los requisitos establecidos basándose directamente en los Casos de Uso, Requisitos o Historias de Usuarios descritas con el objetivo de verificar que los resultados esperados ocurran cuando se usen datos válidos y que se muestren los mensajes de error y precaución cuando se usan datos inválidos. Básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas.

Las pruebas funcionales en la mayoría de los casos son realizadas manualmente, pero esta tarea se torna compleja cuando el sistema que se prueba es demasiado grande y el tiempo para ejecutar todos los casos de pruebas es corto, también es posible automatizar este tipo de pruebas utilizando una herramienta, como es el caso de Selenium IDE que permite la generación automática de casos de prueba creando mayor facilidad en el registro y ejecución de los test, las pruebas realizan exactamente las mismas operaciones cada vez que se ejecutan, de tal modo se elimina en gran medida el error humano. Además las personas generalmente son mucho más lentas que herramientas automatizadas, por tanto la ejecución de las pruebas con herramientas se realiza de manera más rápida reduciendo el número de recursos asociados. La automatización de las pruebas puede resultar compleja y solo es recomendable en algunas funcionalidades específicas, por ejemplo en las pantallas que tendrán mayor uso, generalmente pantallas de ingreso de datos.

Actualmente el interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no responden a sus necesidades, debido a eso las empresas dedican grandes recursos en la definición de estándares y modelos de calidad empleados en la realización de pruebas al software como es el caso de TMMI (Test Maturity Model Integration) o TPI (Test Process Improvement) los cuales aportan un enfoque más amplio y estructurado al proceso de prueba permitiendo visualizar el nivel de madurez del proceso de pruebas y ayuda a definir pasos de mejora en este sentido, pero los resultados alcanzados generalmente no cubren las expectativas de una gran parte de las empresas dedicadas a realizar pruebas al software, el resultado de las pruebas casi nunca cumple con la calidad requerida ya que muchos errores son encontrados en fases posteriores a las pruebas.

Contar con una definición de estrategia de pruebas no es suficiente para la detección oportuna de la mayoría de los defectos en el software. La experiencia en la realización de pruebas a determinado tipo de producto es un aspecto determinante en la fácil identificación de errores comunes y en la agilidad en el momento de la ejecución. Analizar las semejanzas entre las funcionalidades de los sistemas que se someterán a pruebas, es una actividad en la que es muy importante la experiencia de los probadores, pero en los casos de que no todos los probadores cuenten con la experiencia necesaria, este aspecto es más difícil de analizar.

Se hace necesario entonces aplicar herramientas que permitan detectar más defectos en los sistemas contando con la experiencia acumulada en las pruebas a proyectos anteriores similares y formando una base de conocimientos que sirva de apoyo a las pruebas de nuevos proyectos.

1.1 INTELIGENCIA ARTIFICIAL. RAZONAMIENTO BASADO EN CASOS

"La inteligencia artificial es una de las áreas más fascinantes y con más retos de las ciencias de la computación, ya que ha tomado a la inteligencia como la característica universalmente aceptada para diferenciar a los humanos de otras criaturas, ya sean vivas o inanimadas, para construir programas o computadoras inteligentes." (Rodríguez, 2010)

En general, el campo de la IA está orientado a conseguir que las máquinas realicen trabajos donde se aplique la inteligencia, el razonamiento y el conocimiento de un ser humano.

La IA posee técnicas que han sido explotadas en la solución de problemas de ayuda a la toma de decisiones, en el diagnóstico y tratamiento de pacientes por sus potencialidades. Mediante estas técnicas lo que se pretende es emular la capacidad del ser humano al enfrentarse a una toma de decisión, imitando tanto su aprendizaje como la manera de llegar a una decisión basándose en sus conocimientos; características que son las bases para el diagnóstico y el tratamiento, dentro de estas técnicas, se encuentra el RBC que no es más que una manera de razonar haciendo analogías, no sólo es un método poderoso para el razonamiento de computadoras, sino que es usado por las personas para solucionar problemas cotidianos. Se puede concluir que todo razonamiento es basado en casos porque está basado en la experiencia previa.

Por ejemplo en la producción de software, un programador experto que debe arreglar un problema en el código, puede ir directo a donde está el error porque pudo haber arreglado uno anteriormente que tenía los mismos síntomas, dicho programador está aplicando RBC. El RBC constituye un nuevo método de solución de problemas para el desarrollo de sistemas expertos los cuales tienen las numerosas ventajas, ya que el esfuerzo en la solución de problemas puede ser capturado para ahorrar trabajo en el futuro, experiencias previas que hayan sido exitosas pueden ser utilizadas para justificar nuevas, y al contrario experiencias previas que no hayan sido exitosas se pueden utilizar para anticipar problemas. Además la comunicación entre el sistema y los expertos se realiza en base a ejemplos concretos, es decir, el sistema explica sus decisiones citando precedentes.

La característica común del RBC, es el hecho de llevar a cabo una localización aproximada de experiencias previas y una selección de la mejor de ellas en base a la similitud con la nueva situación.

El RBC se puede utilizar para incrementar la calidad del software en el momento que se ejecutan las pruebas funcionales, se pueden obtener los casos de pruebas con los escenarios propicios para obtener defectos de acuerdo a las características del sistema que está bajo pruebas. Estos posibles escenarios se infieren de una base de casos de pruebas realizadas anteriormente a sistemas con requisitos y funcionalidades semejantes. La reutilización de pruebas funcionales mediante los casos de prueba ya diseñados y probados con anterioridad permite una mayor agilidad y confiabilidad en el desarrollo de esta actividad.

2. PRUEBAS FUNCIONALES MEDIANTE EL USO DEL RBC

Un sistema que utilice RBC necesita una serie de experiencias, llamadas casos, almacenadas en una base de casos, donde cada caso se compone generalmente de una descripción del problema y la solución que se aplicó, es posible entonces comparar los casos y adaptarlos de manera efectiva. Los casos se pueden generalizar en alguna medida y es posible abstraer sus características relevantes.

El proceso de la adquisición del conocimiento, con frecuencia es uno de los más difíciles, y se necesita definir cuáles son los principales rasgos de los sistemas que son sometidos a pruebas ya que los problemas tienden a repetirse y, por ello, la experiencia es un recurso útil. Esta propuesta tiene como principales rasgos los requisitos funcionales del sistema.

En la Figura 1 se muestra de forma general los pasos a seguir para usar el Razonamiento Basado en Casos donde Almacenar Información y Solicitar Recuperación de la información son los principales pasos.

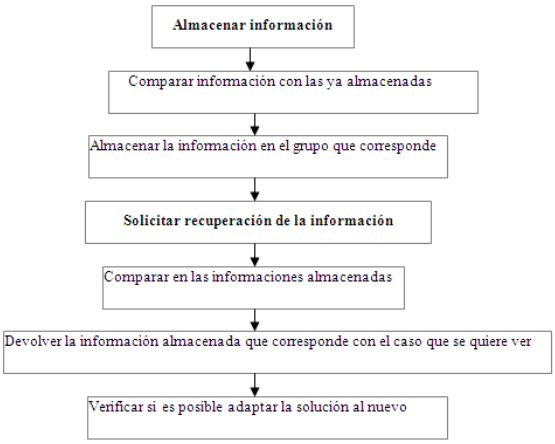

Figura. 1: Pasos del RBC

La utilización del RBC para las pruebas funcionales en los proyectos de software se puede realizar basada en las características particulares de cada proyecto. Al recibir solicitudes de pruebas de varios centros de producción de software, los cuales pueden tener en sus proyectos funciones semejantes y en base a esa semejanza se almacena una buena información para luego usarla en el nuevo proyecto a realizarle pruebas funcionales.

Es necesario realizar la abstracción del conocimiento mediante el apoyo de una red semántica, la cual se representa a continuación:

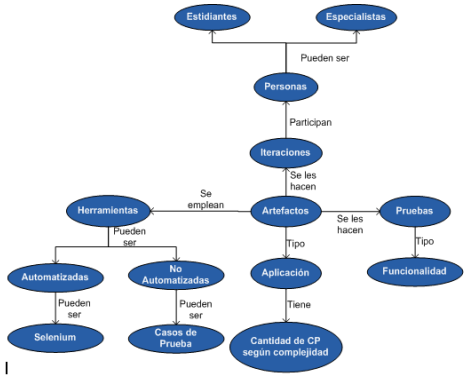

Figura. 2: Red semántica de la base de casos

Luego de tener una mayor noción de cuáles serían las relaciones que tendrán cada rasgo representados en la base de conocimiento, se siguen los siguientes pasos:

- Definir el tipo de requisito por el cual se va a guiar el RBC.
- En caso de ser por los requisitos, reducirlos informalmente de una manera que se pueda comparar en semejanza con otros.
- En caso de ser por proyecto en general, se recupera la información necesaria por el centro al que pertenezca.
- Verificar si es aplicable la información recuperada y realizar el plan de pruebas funcionales.
- Trabajar en las pruebas funcionales en base a dicha información ejecutando los escenarios más propensos a contener errores y verificando los errores más comunes de acuerdo al tipo de requisito que se pruebe.

Importante aclarar que no es necesario trabajar en las pruebas funcionales con toda la información recuperada, sino con las más priorizadas, o sea, las que más reiteración tengan en la base de casos. Todo esto es muy útil para hacer un buen RBC para las Pruebas Funcionales.

Un ejemplo concreto de cómo usar el RBC para las Pruebas Funcionales, específicamente para detectar No Conformidades de Validación es la siguiente.

Según la Figura 1, existen dos momentos importantes:

- Almacenar Información
- Solicitar Recuperación de la información

En el caso del primer momento lo dividiremos en dos más:

- Registro del Proyecto (Rasgos del proyecto que serán usados para establecer las semejanzas)
- Registro del resultado de la prueba

Registro del Proyecto:

Para el primer paso, los parámetros a registrar en dependencia de las características de los proyectos que pasan sus fases de pruebas son los siguientes:

- Nombre del Centro (Nombre textual del centro, en caso de que ya se haya introducido anteriormente ese centro es solo seleccionarlo)
- Nombre del Proyecto e Identificador ID (Nombre textual del proyecto y un identificador, en caso de que ya se haya introducido anteriormente ese proyecto es solo seleccionarlo)
- Características (Algunas característica que distinga al proyecto que pueda servir para compararlo con otro)
- Escoger Tipo de Requisito (Si usa Caso de Uso, Requisitos, Historias de Usuario u otro. Registrar cantidad y nombre de cada uno)
- Escoger Acciones (En dependencia del Tipo y de la cantidad seleccionada anteriormente, registrar por cada una si es un Adicionar, Modificar, Eliminar, Buscar, Reporte o Calcular)
- Escoger Opciones (Las opciones son los tipos de No Conformidades que se encuentran en las Pruebas Funcionales)
    1. Validación
    2. Opciones que no funcionan
    3. Excepciones
    4. Funcionalidad

En esta propuesta solo se abordará la opción de Validación. Existen dos formas de registrar esta información.

- Seleccionar de cada Acción la cantidad de campos que tiene y de cada campo los tipos de campos que existen.
- Seleccionar de cada Acción los tipos de campos que tiene y por cada campo escoger la cantidad.

Los tipos de campos son:

- Campos obligados
- Campos alfanuméricos
- Campos alfabético
- Campos numéricos
- Campos con caracteres especiales
- Campos con números enteros
- Campos con una cantidad exacta de caracteres

Registro del resultado de la prueba:

Luego de estar registrado el proyecto en la Base de Casos, se puede continuar con el presente caso o solicitarlo posteriormente haciendo los pasos siguientes:

- Buscar proyecto (Se realiza una búsqueda de los proyectos usando el ID que se introdujo)
- Listar los Casos de Uso o Requisitos o Historias de Usuario u Otros.
- Listar las Acciones de cada Tipo listado anteriormente.

- Escoger la Opción Validación (En este caso específico se verá solo Validación)
- Listar cada campo introducido (Este caso solo es válido si escogió la opción 1 anterior)
- De cada campo introducido, listar el/los tipos de campos.
- De cada tipo de campo, registrar el resultado de la prueba.

La forma en que se debe registrar el resultado de la prueba sería seleccionar Satisfactoria o Insatisfactoria, donde Satisfactoria es que no tuvo No Conformidades en ese campo e Insatisfactoria lo contrario y así es más fácil realizar las comparaciones pertinentes para seleccionar los posibles errores en un proyecto con características semejantes.

Solicitar Recuperación de la información:

Luego de hacer el paso Registro del Proyecto, es posible realizar los pasos siguientes para devolver los posibles errores que puede tener ese proyecto.

- Comparar con el Tipo escogido anteriormente (Dígase Caso de Uso, Requisitos, Historias de Usuario u otro) y escoger las Acciones (Dígase Adicionar, Modificar, Eliminar, Buscar, Reporte o Calcular) y Opciones (En este caso es sólo Validación) que más semejanzas presenta teniendo en cuenta una cantidad mínima de coincidencia.
- Devuelve los posibles errores (Registro del resultado de la prueba)

Hasta aquí se describen las principales actividades para el registro de los rasgos significativos de los sistemas a los cuales se le ejecutarán pruebas funcionales y la forma en que se recuperaría la información del resultado de la prueba que se puede inferir en el resultado para una nueva prueba de funcionalidad. Todos estos rasgos se deben almacenar en una base de casos para posteriormente inferir los posibles resultados en las pruebas mediante un Sistema Experto.

3. SISTEMAS EXPERTOS

Según Turban y Aronson, un sistema experto es un sistema que utiliza conocimiento humano capturado en una computadora, para resolver problemas que ordinariamente requieren el expertise humano. (Ignacio, 1991)

De acuerdo con la manera que se representa el conocimiento en la base de conocimiento y por consiguiente la forma de funcionamiento del mecanismo de inferencia, existen diferentes tipos de Sistemas Expertos. El sistema que se aplicará a esta investigación es del tipo basado en casos.

Los sistemas expertos son generalmente costosos, pero el mantenimiento y el precio de su uso repetido es respectivamente bajo. Por otra parte, los ingresos en términos monetarios, tiempo, y exactitudes resultantes del uso de sistemas expertos son considerablemente altos. Sin embargo, antes de desarrollar o adquirir un sistema experto se debe efectuar un análisis de las posibilidades que el sistema va a brindar. Hay varias razones para utilizar sistemas expertos, las más significativas son: (Moreno, 2011)

- Con la ayuda de un sistema experto, una persona con poca experiencia puede solucionar problemas que requieren conocimiento de expertos. Esto es también importante en casos en los que hay pocos expertos humanos. Además, con el uso de sistemas expertos el número de personas con acceso al conocimiento aumenta.
- Con la combinación de los conocimientos de varios expertos humanos, puede dar lugar a sistemas expertos más seguros, ya que se obtiene un sistema experto que combina el conocimiento colectivo de varios expertos humanos en lugar de la de uno solo.
- Los sistemas expertos pueden responder a preguntas y resolver problemas con mucha más rapidez que un experto humano. Por ello, los sistemas son muy valiosos en casos en los que el tiempo de respuesta es escaso.
- En algunos casos, la complejidad del problema impide al experto humano resolverlo. Debido a la capacidad de los ordenadores de procesar un elevadísimo número de operaciones a gran velocidad, los sistemas expertos suministran respuestas rápidas y fiables en situaciones en las que los expertos humanos no pueden.

- Los sistemas expertos pueden ser utilizados para realizar operaciones monótonas, aburridas e inconfortables para los humanos.

Los Sistemas Expertos no basan su resolución en el mismo principio, es por ello que antes de elegir cual se utilizará, se deben estudiar las características del área del conocimiento donde el Sistema Experto será el protagonista, pues todos no utilizan la misma analogía y a la hora de inferir las soluciones recurren a técnicas específicas en cada caso.

3.1 SISTEMA SI-HOLMES

Todos los Sistemas Expertos no son iguales, y no basan su resolución en el mismo principio, es por ello que antes de elegir cual se utilizará, se deben estudiar las características del área del conocimiento donde el Sistema Experto será el protagonista, pues todos no utilizan la misma analogía y a la hora de inferir las soluciones recurren a técnicas específicas en cada caso.

El SI-Holmes es un Sistema Experto desarrollado en la UCI, el cual está diseñado para temas generales ya que se puede utilizar en múltiples ramas, desde la rama jurídica hasta la rama de desarrollo de software. A continuación se exponen diferentes características de dicha herramienta.

Esta aplicación es producto de una investigación realizada para las mejoras en las planificaciones de las pruebas de software. Escrito en el lenguaje Java muy extendido que encierra su fuerza en su independencia a la plataforma, seguridad, portabilidad, escalabilidad, que promueve la reutilización de código. Es una aplicación Web, con una arquitectura 3 capas (Capa de Datos, Negocio e Interfaz) lo cual le provee algunas ventajas como la robustez debida al encapsulamiento, la facilidad de mantenimiento, soporte y flexibilidad, así como alta escalabilidad.

Este sistema posee una interfaz amigable y configurable como se muestra en la Figura 3. Tiene la ventaja de interactuar al mismo tiempo con múltiples bases de caso que se encuentran persistidas en una base de datos, muestra sus nombres, cantidades de rasgos y casos de cada una de ellas.

Figura. 3: Interfaz del SI-Holmes

La calidad de los resultados depende, sobre todo, del cuidado que se ponga en la entrada de los datos a la base de conocimiento y que exista una buena cantidad de casos con todas las posibles combinaciones que puedan existir, ya que esto influye decisivamente en la exactitud de los resultados.

4. CONCLUSIONES

En la anterior investigación se propone una base de casos que permita inferir los principales errores detectados durante las pruebas funcionales a partir de rasgos semejantes de sistemas probados previamente.

Se propone la herramienta SI-Holmes como Sistema Experto para inferir automáticamente los resultados de las pruebas y evitar en gran medida el error humano en la no ejecución de importantes escenarios de pruebas que comúnmente puedan contener errores.

La solución presentada, que permite aumentar la calidad y eficiencia de las pruebas funcionales utilizando el RBC representa una significativa ayuda para las empresas que se dediquen a evaluar la calidad de un sistema, especialmente para aquellas que no cuenten con probadores expertos expertos ni el tiempo necesario para realizar detalladamente todas las pruebas en escenarios posibles.

REFERENCIAS

Ignacio, J. (1991). "*An Introduction to Expert System, MgGraw-Hill Computer Science"*.

NC ISO/IEC 9126-1 (2003). *Tecnología de la información. Características de calidad y métricas del software.* Oficina Nacional de Normalización, La Habana.

Rodríguez, M.G. (2010). *"Sistema Experto para el diagnóstico médico de las enfermedades genéticas con dismorfias (SEGEDIS)"*. La Habana

Roger S, P (2010). *"Ingeniería de Software. Un enfoque práctico"*. La Habana

Yenier, M. (2011). *"Base de conocimientos para inferir el comportamiento de las pruebas de liberación en el Laboratorio Industrial de Pruebas de Software"*, Tesis de grado, Universidad de las Ciencias Informáticas, Ciudad de La Habana

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.