ISSN 1870-4069

Ingeniería inversa en base a pruebas unitarias

L. Alberto Nicolás Ramírez[1], Carlos Perez-Corona[1], Yesenia N. González-Meneses[2]

Instituto Tecnológico de Apizaco, Departamento de sistemas y computación,
Tlaxcala, México

Facultad de Ciencias Básicas, Ingeniería y Tecnología, UAT,
Tlaxcala, México

{nic.lui.ram, cperezcorona}@gmail.com, yeseniaglez@hotmail.com

Resumen. Las metodologías proporcionan herramientas para llevar a cabo el correcto desarrollo de software buscando obtener software de calidad, en este sentido la base documental que describe a un sistema juega un papel importante al momento de dar mantenimiento, por ejemplo. Sin embargo, por malas prácticas es común encontrar sistemas sin esa base documental. Es allí donde surge la necesidad de crear una metodología que permita generar los productos base de la ingeniería de software. Este trabajo se enfoca a resolver el problema antes mencionado partiendo de pruebas unitarias utilizando software de testing y la interacción del usuario con el sistema o plataforma de caso, obteniendo un panorama general con los productos obtenidos, los diagramas UML obtenidos modelaran el sistema para ser comprendido e inferir los requerimientos iniciales, así como encontrar posibles errores, como pueden ser bugs, objetos o métodos no utilizados y detectar posibles riesgos generando problemas internos en la ejecución del sistema.

Palabras clave: Prueba del software, ingenieria de software, prueba de unidad, ingenieria inversa.

Reverse Inverse Engineering Based on Unit Tests

Abstract. The methodologies provide tools for the proper development of software with quality, in this sense the documental base that describes a system plays an important role at the moment of maintenance, for example. However, due to bad practices, it is common to find systems without this documental base. From here, arises the need to create a methodology that allows generating the base products of software engineering. This work focuses on solving the aforementioned problem starting from unit tests using testing software and the user's interaction with the system or platform of the case, the UML diagrams obtained will model the system to be understood and infer the requirements. As well as to find possible errors, such as bugs, unused objects or methods and to detect possible risks generating internal problems in the execution of the system.

Keywords: Software testing, software engineering, unit test, reverse software engineering.

1. Introducción

El desarrollo de software, se divide en fases que contemplan el análisis de requerimientos buscando conocer las necesidades del cliente, en base a ellas se modela una solución y posteriormente entrar a la fase de codificación, continuando así a la fase de pruebas; por cada fase debe generarse productos documentales. La fase de pruebas debe ser considerada por el equipo de desarrollo para asegurar la integridad del software cuando se encuentre en producción.

Este trabajo propone una metodología utilizando ingeniería inversa a partir de pruebas unitarias.

Para iniciar la metodología analiza las funciones del sistema (o plataforma) obteniendo los componentes claves que la integran y generando una base documental que ayudará a establecer parámetros de evaluación. Contando con los componentes de la plataforma se debe establecer las herramientas de evaluación idóneas para la ejecución de pruebas mediante frameworks de testing.

Habiendo realizado pruebas se obtendrán los objetos y actores que intervienen generando entonces los diagramas de casos de uso.

En base a los diagramas de caso de uso puede generarse los diagramas de clases y de secuencia.

Con la información obtenida se generará un documento mostrando los resultados obtenidos de las pruebas unitarias resaltando fallas, observaciones y métodos no utilizados; dicho documento será la referencia para resolver dichos problemas.

Por último, serán validados los requerimientos iniciales de la plataforma y se volverá a la parte inicial de la metodología o terminar el ciclo e validación.

2. Estado del arte

Software testing for object-oriented software.

La propuesta descrita en [1] busca mostrar los pasos para realizar pruebas de manera correcta, mostrando qué diagramas son necesarios para realizar dichas pruebas.

La creación de diagramas UML es el fuerte de esta propuesta, ya que enfatiza la necesidad de ellos para dar mayor precisión a los resultados.

Sin embargo no se busca generar documentación solo explica una sugerencia de pasos a seguir para hacer pruebas al software.

Attributes effecting software testing estimation; is organizational trust an issue?

La propuesta en [2] busca implementar correctas prácticas para el desarrollo de pruebas mediante reuniones para establecer dichos parámetros.

Si, bien las reuniones del equipo de software juegan un papel importante, esta propuesta hace notoria la necesidad de las mismas, esto con el objetivo de que el equipo

de desarrollo esté al tanto de los avances y estatus del proyecto y con esto disminuir la presencia de errores considerablemente.

La propuesta expresa como establecer métricas para evaluar software, sin embargo no busca generar diagramas de modelado del sistema.

A framework of the use of information in software testing. En [3] se busca entrar más en materia acerca de las pruebas de software, aquí se menciona la necesidad de realizar entrevistas al equipo de desarrollo para poder generar pruebas más exactas y comprobar la calidad del producto. Contar con información del software a desarrollar es importante para garantizar que el software se hará como fue solicitado y con eso evitar retrasos o trabajo de más en el desarrollo del mismo. Se menciona la generación de productos, pero no busca analizar el código fuente y solo se centra en pruebas unitarias.

3. Metodología

La propuesta de metodología que se muestra en la figura 1, se enfoca al análisis del algún software o sistema de información que no cuente con base documental, y que mediante una serie de pruebas unitarias pueda determinar los requerimientos iniciales y los productos de ingeniería de software del sistema, podrá verificarse si dicho sistema fue diseñado o codificado de acuerdo a los requerimientos. La metodología busca obtener los siguientes objetivos:

- Generar una metodología que en base a pruebas unitarias.
- Evaluar la plataforma utilizando pruebas unitarias.
- Identificar ventajas de frameworks de testing para evaluar la plataforma.
- Generar productos de las fases de análisis, diseño e implementación a partir de pruebas.
- Identificar a los actores que intervienen en la plataforma.
- Documentar de casos de uso.
- Documentar las pruebas expresándolas en diagramas de caso de uso.
- Representar la relación de los objetos entre si mediante diagramas de secuencia.
- Mostrar la estructura estática del sistema mediante diagramas de clases.
- Identificar posibles debilidades en la plataforma.
- Dar opciones de mejora en el desarrollo de software.

Análisis de la plataforma a evaluar para la comprensión de la misma. En esta fase se analizara el estado actual del sistema, mediante la observación y la interacción con la misma. Para ello es necesario entrevistar a quien interactúa con la plataforma a manera de conocer cómo funciona. Es necesario que el analista también interactúe con la plataforma para obtener una visión general del funcionamiento de la misma. Con una serie de preguntas y respuestas hechas por el analista al equipo de trabajo que utiliza la plataforma.

Análisis de frameworks adecuados para la evaluación. En esta fase es necesario determinar que Frameworks de testing que existen y en base al lenguaje de la

plataforma. Con ello se puede establecer el tipo de pruebas y obtener un mayor conocimiento del funcionamiento de la plataforma e identificar los actores y casos de uso de la que intervienen en la plataforma.

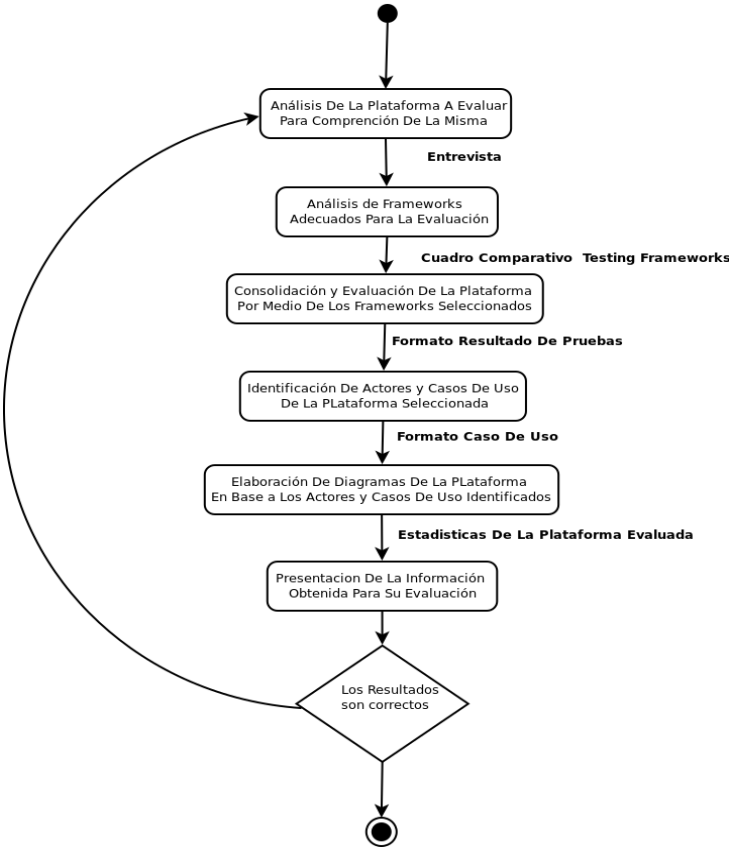

Fig. 1. Metodología de Ingeniería inversa en base a pruebas unitarias.

Consolidación y evaluación de la plataforma por medio de los frameworks seleccionados. En esta fase es deseable tener acceso al código fuente y la interfaz gráfica de la plataforma y analizando la función de cada uno de los componentes del sistema. Realizando pruebas unitarias para comprobar, si cada uno de los componentes del sistema realiza la función deseada. Por ejemplo el método A espera un valor entero resultado de una consulta, se debe poner a prueba dicho método ya sea enviando valores nulos o algún entero que sea igual a 0.

Identificación de actores y casos de uso de la plataforma seleccionada. En esta fase se ha adquirido un mayor conocimiento del funcionamiento y estructura de la plataforma seleccionada y es donde es necesario identificar a los actores que intervienen, estos pueden ser el usuario del sistema y pueden derivarse más actores, si

la plataforma seleccionada cuenta con roles de usuario, por ejemplo administrador e invitado.

Elaboración de diagramas de la plataforma en base a los actores y casos de uso identificados. En esta fase se tendrá una visión general del funcionamiento del sistema para generar diagramas de caso de uso e identificar como los usuarios (actores) y las acciones que realizan (casos de uso). Sin embargo es necesario demostrar la secuencia detallada de los diagramas de caso de uso, por lo que será necesario recurrir a los diagramas de secuencia por ultimo para mostrar la estructura estática del sistema se deben realizar diagramas de clases.

Presentación de la información obtenida. Última fase, donde en base a los diagramas y el resultado de las pruebas unitarias se puede inferir cuales son los requerimientos iniciales cotejando con la plataforma actual si estos son los correctos, obteniendo una visión general de la estabilidad y funcionalidad de la plataforma y poder dar propuestas de mejora si las hubiera.

4. Demostración de la metodología

Análisis De La Plataforma A Evaluar. Dentro de este apartado se mostrará los resultados del análisis de la plataforma utilizada como caso de estudio, que se encuentra en fase de producción, sin embargo, aun no se han realizado las pruebas pertinentes para evaluar la calidad del mismo. Considerando que no existe una base documental ni productos de ingeniería de software, no existe certeza que el resultado de las pruebas sea el esperado; por lo tanto fue necesario entrevistar al equipo de desarrollo y directivos de la empresa para que la información permita determinar que pruebas miden o evalúan la calidad de la plataforma.

Análisis de frameworks Adecuados para la evaluación. En esta fase se muestran los frameworks que serán utilizados para evaluar la plataforma. A continuación serán explicadas las ventajas y desventajas de los frameworks seleccionados:

Arquillian. Mediante el objeto **ShrinkWrap** permite construir el tipo de empaquetado del proyecto que se esté utilizando (.jar, .war etc.) y ejecutar el test, está característica permite trabajar en conjunto con servidores de aplicaciones como GlassFish o JBoss [5] y utilizar el criterio de persistencia para realizar conexión a un servidor de base de datos como mysql [6] o SQL Server [7] *Arquillian* utiliza librerías de JUnit y tiene soporte con JPA y se puede contemplar como una opción viable para implementar el testing.

Selenium. Este framework guarda los datos que fueron digitados al realizar dichas pruebas. *Selenium* registra el evento y datos digitados por el usuario y los valores que fueron digitados o generados por el evento se registran en la interfaz del programa.

Consolidación y evaluación de la plataforma por medio de los frameworks seleccionados. Los criterios de las pruebas efectuadas a la plataforma serán mostradas en esta sección.

Test Unitario: Arquillian permite crear instancias objetos de clase y por ende invocar sus métodos, de tal modo que sea posible realizar un "deploy" de la clase en forma aislada resultando en una validación de resultados y comportamientos.

Niveles de Pruebas. Las pruebas realizadas en la plataforma se dividen en dos fases, unitarias, funcional e integración, esto con la finalidad (o propósito) de realizar un análisis adecuado. [4].

Test Funcional e Integración: Se reproducirán las pruebas grabadas previamente con una serie de parámetros diferentes con la finalidad de evaluar la comunicación de la interfaz gráfica (front end) con los componentes de más bajo nivel (back end).

Parámetros de Evaluación. Para determinar sí las pruebas realizadas a la plataforma cumplen con los criterios esperados, garantizando la calidad de los componentes, se realizan evaluaciones en base a la salida obtenida y avalados por los líderes de proyecto.

Dichos parámetros son formados por los siguientes elementos:

- Reunir los datos básicos de un caso de uso.
  - Nombre Del Requisito Funcional
  - Versión
  - Objetos Asociados
- Descripción
  - Pre-Condición: mostrará qué acción debe ser invocada o ejecutada para utilizar el caso o casos de uso.
  - La secuencia de los eventos y qué casos de uso intervienen.
  - Post-Condición: mostrará el resultado esperado cuando se termine la ejecución del caso de uso.
  - Excepciones
  - Rendimiento
  - Importancia
  - Urgencia
  - Comentarios

Identificación de actores y casos de uso de la plataforma seleccionada. En esta fase se obtienen los actores que intervienen en la plataforma. En nuestro ejemplo, los actores identificados son los de la tabla 2.

Casos de uso. De la entrevista realizada al equipo de desarrollo se obtienen las funciones de la plataforma. Siendo identificados casos de uso y actores se procede a crear los diagramas UML para modelar la plataforma, en este caso sólo se explicará con el resultado de un diagrama de casos de uso: Administración de Usuarios.

Tabla 1. Identificación de actores.

Nombre Actor	Descripción	Tipo
Usuario Administrador	Usuario que cuenta con el acceso a todas las funciones de la plataforma	Persona
Usuario General	Representación de las funciones que, los usuarios comparten en común	Persona
Sistema	plataforma	Sistema

Elaboración de diagramas en base a los actores y casos de uso identificados. En esta fase se mostrará los diagramas de casos de uso, de clases y secuencia generados. En base a los diagramas de caso de uso generados que fueron analizados se tiene como resultado obtener diagramas de secuencia del caso diagrama anterior generando diagramas de secuencia y diagramas de clase, dichos diagramas modelaran el módulo o función analizada.

Para un mejor entendimiento se explicara el diagrama utilizando el patrón de diseño MVC (Modelo-Vista-Controlador). A continuación se explica un módulo del sistema analizado.

Administrador de Usuarios. Una de las funciones principales del sistema es el módulo de usuarios, donde se muestra quien puede acceder al sistema y los permisos con los que cuenta. Este módulo permite crear, editar, eliminar y consultar los usuarios registrados, para acceder a este módulo es necesario contar con permisos de administrados o supervisor. De donde se obtuvieron los diagramas que se ilustran en las figuras 2, 3 y 4. La figura 2 muestra cómo se obtuvieron los casos de uso utilizando la información contenida en el formato que se describe en la figura 3.

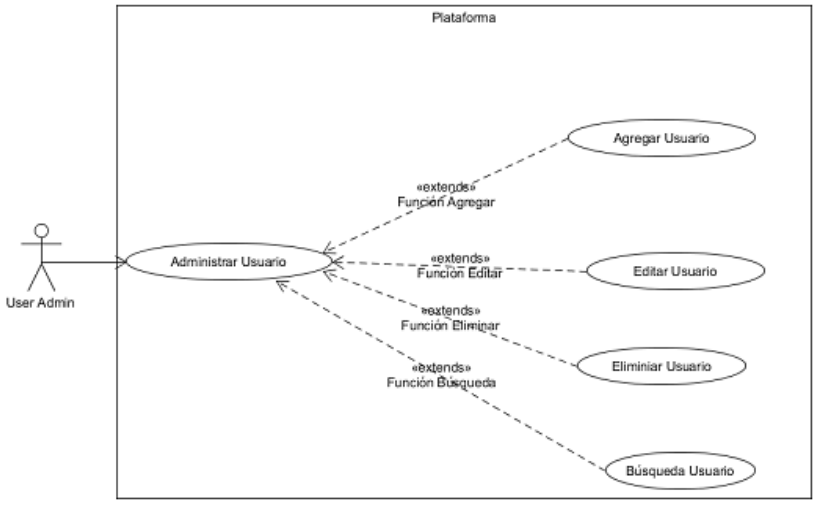

Fig. 2. Diagrama caso de uso administrador de usuarios.

RF- <id del requisito>	User-Admin.xhtml	001
Versión	0.3	
Autores		
Fuentes		
Objetivos asociados	Index.xhtml	
Descripción	Administración de usuarios, muestra los datos básicos de los usuarios y permite editar, agregar y asignar usuarios.	
Precondición	Haber iniciado cuenta de administrador.	
Secuencia Normal	**Paso**	**Acción**
	1	El *usuario* **selecciona** *(load)*el módulo de usuarios.
	2	El *sistema* **muestra** (*readAll)* los usuarios activos registrados.
	3	El *usuario* **selecciona** <<función>>
	4	El *sistema* **ejecuta** <<función>>
Pos condición	Guardar cambios realizados	
Excepciones	**Paso**	**Acción**
	2.1	El sistema no muestra resultados por error de conexión
	4.1	El sistema no ejecuta la función seleccionada.
Importancia	Vital	
Urgencia	Urgente	
Casos de uso extensión	Add-User Edit-User Delete-User Busqueda-User	

Fig. 3. Formato de caso de uso administrador de usuarios.

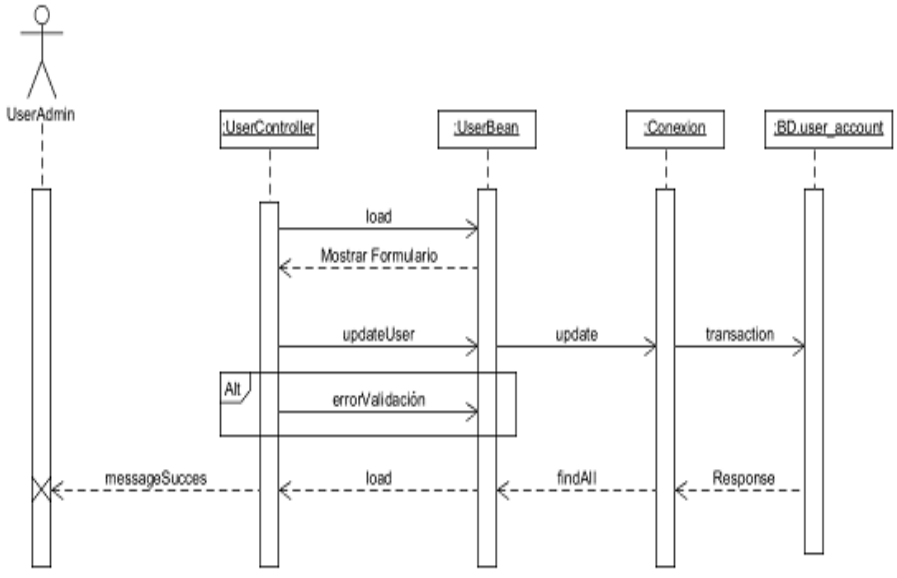

Fig. 4. Diagrama de secuencia administrador de usuarios.

Presentación de la información obtenida. Esta fase muestra al equipo de desarrollo la información obtenida del caso de uso (User-Admin) es la que se presenta en la tabla 2, esta información debe ser utilizada para mejorar la plataforma o sistema, la inferencia de los requerimientos se dan al analizar los diagramas obtenidos.

Productos obtenidos. Al realizar la metodología se obtienen los productos mostrados en la tabla 3, los cuales son necesarios para inferir los requerimientos iniciales o buscar errores y opciones de mejora.

Tabla 2. Información obtenida.

Observaciones	Errores	Total
Métodos que no manejan excepciones	161	346
Métodos que retornan null alguna query	221	346
Errores y observaciones encontradas	53	346
Métodos no utilizados	100	346

Tabla 3. Productos obtenidos.

Fase	Producto
Análisis de la plataforma a evaluar para la comprensión de la misma	Entrevista equipo desarrollo
Análisis de frameworks,adecuados para la evaluación	Cuadro comparativo de frameworks testing
Consolidación y evaluación de la plataforma por medio de los frameworks seleccionados	Formato resultado de pruebas
Identificación de actores y casos de uso de la plataforma seleccionada	Formatos de caso de uso
Elaboración de diagramas de la plataforma en base a los actores y casos de uso identificados	Estadísticas de la plataforma evaluada
Presentación de la información obtenida	

5. Conclusion y trabajos futuros

La ejecución de pruebas para el desarrollo de software es imprescindible, ya que la parecencia de errores en la fase de producción puede presentar un retraso significativo en la entrega del producto. Inicialmente se requería tener interacción con la plataforma y el equipo de desarrollo, debido a que era necesario conocer cómo funcionaría la plataforma, posteriormente se diseñaron pruebas con un framework de testing previamente analizadas donde se comprobó la funcionalidad de dicha plataforma. Como resultado de las pruebas unitarias se generaron productos de ingeniería de software como: actores, diagramas de caso de uso, diagramas de clase, diagrama entidad-relación y diagramas de secuencia. Terminada la metodología se mostraron los resultados al equipo de desarrollo haciendo propuestas de mejora si las hubiese.

Trabajos futuros. Para la mejora del trabajo antes expuesto es necesario implementar nuevas características a la metodología, las cuales serán mencionadas:

- Crear más diagramas UML para dar mayor exactitud a la metodología y que estos cambien en función a la arquitectura de la plataforma a evaluar.
- Adaptar la metodología para que sea compatible con la reingeniería de software, que en base al diseño de pruebas pueda ser interpretada como ingeniería inversa.
- En base al acceso que se requiere al código fuente se necesita hacer propuestas de optimización de código, por ejemplo, hacer reducción e identificación de variables públicas o privadas que no sean utilizadas y que puedan generar fugas de memoria.
- De acuerdo al patrón de diseño con el que fue diseñada la plataforma permitir la adaptación de la metodología.
- Implementar pruebas de sistema e integración o diseñar pruebas propias para dar mayor exactitud al resultado final.
- Permitir que la metodología sea compatible con mas arquitecturas de software.
- Suite dinámica que en base a la interacción del equipo de desarrollo genere los diagramas de la metodología.

Referencias

1. Biju, S. M.: Model-based software testing for object-oriented software. E-Learning, no. 5, pp. 885–891 (2008)
2. DuBois, P.: Mysql 5.0 reference manual. Disponible en: http://dev.mysql.com/ doc/refman/ 5.0/en/index.html (2015)
3. Hammoud, W.: Attributes effecting software testing estimation; is organizational trust an issue? Phoenix, ProQuest LLC, D.M.IST, Dissertation, University of Phoenix (2014)
4. it Mentor: Capacitación y guía para el desarrollo de software. Disponible en: http://materias.fi.uba.ar/7548/PruebasSoftware.pdf (2015)
5. JBoss: About jboss. Disponible en: http://www.jboss.org (2015)
6. Microsft: Sql server. Disponible en: http://www.microsoft.com/es-es/server-cloud/products/sql-server (2015)
7. Payman, K.: A framework of the use of information in software testing. Proquest LLC, D.Sc., Dissertation, The George Washington University (2010)