PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DE UNA INFRAESTRUCTURA DE SOFTWARE PARA REALIZAR PRUEBAS AUTOMATIZADAS DE SISTEMAS DE INFORMACIÓN DESARROLLADOS EN LENGUAJE COBOL EN EL CONTEXTO BANCARIO.

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Kenjy Tasato Cánepa

Código: 20040319

ASESOR: Abraham Eliseo Dávila Ramon

Lima, Julio del 2013

Resumen

El presente proyecto de fin de carrera corresponde al desarrollo de una infraestructura de software para realizar pruebas automatizadas de sistemas de información desarrollados en lenguaje cobol en el contexto bancario.

La propuesta de esta infraestructura surge como resultado del análisis de la realidad del proceso de pruebas dentro del área de Certificación de un banco. La infraestructura propuesta es configurable, escalable y adaptable para las diferentes versiones de cobol con las que se cuenten.

El presente documento ha sido estructurado en 7 capítulos como se describe a continuación:

En el capítulo 1 se presenta la problemática de donde se extrajo el problema que tuvo como resultado la propuesta del presente proyecto de fin de carrera, el objetivo general, objetivos específicos y resultados esperados. Se detallan las metodologías utilizadas en el proyecto tanto metodologías del proceso de construcción como la gestión del proyecto. Finalmente se menciona el alcance y limitaciones.

En el capítulo 2 se presentan los conceptos y definiciones escenciales para el desarrollo del proyecto.

En el capítulo 3 se revisan las soluciones actuales al problema identificado, se mencionan herramientas y finalmente se hace una comparación entre ellas.

En el capítulo 4 se mencionan los requerimientos funcionales y no funcionales de la infraestructura.

En el capítulo 5 se presenta la arquitectura de la solución, así como diagramas de actores entre otros, que permitirán comprender la forma de la infraestructura.

En el capítulo 6 se presenta las distintas funcionalidades de la infraestructura ayudado de capturas de pantalla.

En el capítulo 7 se presentan las observaciones, conclusiones y recomendaciones del presente proyecto de fin de carrera.

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **DESARROLLO DE UNA INFRAESTRUCTURA DE SOFTWARE PARA REALIZAR PRUEBAS AUTOMATIZADAS DE SISTEMAS DE INFORMACIÓN DESARROLLADOS EN LENGUAJE COBOL EN EL CONTEXTO BANCARIO.**

ÁREA: INGENIERÍA DE SOFTWARE

PROPONENTE: Abraham Dávila

ASESOR: Abraham Dávila

ALUMNO: KENJY TASATO CANEPA

CÓDIGO: 20040319

TEMA N°: _____

FECHA: Miércoles, 15 de Mayo del 2013

DESCRIPCIÓN

Los bancos son entidades del sector financiero cuyos sistemas informáticos constituyen uno de sus principales pilares sobre el cual se desarrolla toda la operación, siendo la banca por internet la mayor expresión del uso de las tecnologías de información en este sector. Para este tipo de organizaciones, la confianza que recibe del cliente es muy importante para su continuidad por lo que la calidad de estos sistemas informáticos juegan un papel determinante en la imagen del banco que deben mantener o mejorar en el tiempo.

De otro lado es conocido que la mayoría de sistemas bancarios informáticos en el mundo utilizan productos que tienen muchos años operando en el mercado y que los bancos lo adquieren y lo adecuan a su propia organización. Estos sistemas informáticos suelen ser grandes y se encargan de casi toda las operaciones de los bancos y deben ser actualizados de manera continua sea por necesidades de adaptación al marco regulatorio, necesidades propias de nuevas funcionalidades del banco o por detección de defectos.

En particular, el banco que se tomará como referencia tiene un área de Certificación de software orientado a las pruebas funcionales de los sistemas de información críticos que solamente utilizan entornos semejantes a los utilizados para el desarrollo del producto (muchos de ellos en Cobol) y realizando el proceso de pruebas de manera manual. Sin

embargo, la carga de trabajo ha puesto al área de Certificación como un cuello de botella, siendo necesario contar con herramientas que apoyen su productividad y calidad.

Por lo antes descrito, se propone desarrollar una infraestructura de software de modo que se pueda automatizar las pruebas de sistemas de información que están desarrollados en Cobol.

OBJETIVO GENERAL

El objetivo general es desarrollar una infraestructura de software para realizar pruebas automatizadas de sistemas de información desarrollados en Lenguaje COBOL en el contexto bancario.

OBJETIVOS ESPECÍFICOS

Los objetivos específicos son:

- OE1: Caracterizar una infraestructura de software para la gestión y realización de pruebas automatizadas.
- OE2: Definir una arquitectura que se integre de una forma no invasiva al producto y permita una gestión automatizada de distintos casos de prueba.
- OE3: Construir con tecnología adecuada al sistema de información al cual se tiene que integrar en el piloto.
- OE4: Realizar un piloto de la infraestructura de pruebas.

ALCANCE

El sistema informático que se propone desarrollar permitirá hacer una infraestructura para la automatización de pruebas de sistemas informáticos existentes en Cobol; para ello se desarrollara un componente que permita (i) el registro de casos de pruebas que se puedan parametrizar con el objetivo de tener flexibilidad de cambiarlos en distintas ejecuciones de pruebas, (ii) la administración de conjuntos (set) de pruebas como crearlos, agregar o eliminar prueba asociados del conjunto, (iii) la administración de la ejecución de los casos de prueba y de los resultados obtenidos; y (iv) la generación de reportes para apoyar la gestión del personal de pruebas de software. Para validar la infraestructura se desarrollará un sistema reducido en Cobol de modo que se permita comprobar las facilidades ofrecidas por la infraestructura de pruebas.

A Dios por darme perseverancia y siempre guiar mi camino

A mi padre por enseñarme que siempre

puedo alcanzar lo que me propongo

y pese a que no es de muchas palabras

darme fuerzas con un fuerte abrazo

A mi madre por cambiar los malos días con su sonrisa

que lo ilumina todo

A mi hermano por ser mi mejor amigo y consejero

A todos por darme su amistad, cariño y buenas vibras

A mi gran amigo Richard que sin su sabia guía

hubiese estado a la deriva

Agradezco de corazón a Dios, mi familia, mis amigos y sobre todo a mi asesor

Por su perseverancia, paciencia, guía,

y apoyo incondicional para la realización

de este proyecto de fin de carrera

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

CAPÍTULO 1: Generalidades

En este capítulo se presenta la propuesta del proyecto incluyendo la problemática, el marco teórico, el estado del arte entre otros.

1.1.- Introducción

El presente proyecto de fin de carrera está orientado al ámbito bancario y busca desarrollar una solución que podría ser usada en la automatización de las pruebas de sistemas bancarios de larga vida.

De la experiencia previa desarrollada en el sector bancario se sabe que los sistemas informáticos de larga vida, debido a las adecuaciones a normativas y a las presiones del mercado, están en continuas modificaciones. Estos, al ser sistemas (por lo general transaccionales) que han acompañado al banco por largo tiempo, suelen estar desarrollados con lenguajes de programación también antiguos. Los cambios, como parte del proceso de mantenimiento, conllevan a probar el sistema para garantizar que este funcione de la manera correcta y los cambios que se hallan implementado funcionen de manera exitosa sin afectar la funcionabilidad de los componentes impactados (es decir que no tengan impactos colaterales) .Esto implica, por lo general, realizar las mismas pruebas reiteradas veces, no necesariamente en el mismo periodo. Por ejemplo: cuando se cambió la tasa de ITF (impuesto a las transacciones financieras), pese a que solo fue un parámetro el que cambiaba, para asegurar que el sistema bancario funcionaba correctamente fue necesario probar el funcionamiento de todo el sistema informático que administra las transacciones ya que el ITF impacta en prácticamente todas las transacciones de depósitos, transferencias y retiros. Se debe resaltar que lo que se realizó fue una prueba de regresión la cual consiste en probar que el cambio no haya afectado a otros componentes o funcionalidades distintos al que ha sido modificado).

En estos sistemas la tolerancia a errores, por parte del negocio es cero ya que de ello depende la economía del banco y la confianza por parte de los clientes de tener un sistema robusto y sin errores. Debido a esta necesidad es común realizar pruebas de

regresión de manera repetitiva para verificar que las funcionalidades o componentes que no han sido cambiados funcionen de manera correcta.

Sin embargo las pruebas de regresión, así como las pruebas funcionales, unitarias y de sistema, se realizan de manera manual en al menos tres bancos donde se recoge esta experiencia. Lo que sí tienen es una herramienta para administrar los recursos ("testers") por prueba realizada o registro de los casos de prueba pero no su automatización.

El presente proyecto de fin de carrera tiene como fin desarrollar una infraestructura de soporte de pruebas de regresión para sistemas bancarios de larga vida. Esta infraestructura no es sino una estructura construida alrededor del sistema que se va a probar, la cual nos va permitir realizar pruebas sobre el sistema objetivo ya que el sistema a probar no cuenta con un soporte nativo para realizar pruebas.

Una solución de este tipo busca contribuir a la reducción de la carga operacional repetitiva del proceso de pruebas de regresión y a la gestión y ejecución de casos de pruebas. Cabe mencionar que las pruebas que se realizan son a nivel funcional (ello significa comprobar que el sistema en su totalidad o parte de él funciona de manera correcta) de lo que se desprende que se probará una funcionalidad completa y no un "componente" o "método" únicamente.

1.2.- Problemática

Los bancos son entidades del sector financiero cuyos sistemas informáticos constituyen uno de sus principales pilares sobre el cual se desarrolla toda la operación, siendo la banca por internet la mayor expresión del uso de las tecnologías de información en este sector. Para este tipo de organizaciones, la confianza que recibe del cliente es muy importante para su continuidad por lo que la calidad de estos sistemas informáticos juegan un papel determinante en la imagen del banco que deben mantener o mejorar en el tiempo [IBM, 2013]. Como se aprecia en la Figura 1.1 el "50% es el porcentaje de clientes que daría solo dos oportunidades de error a su banco antes de pensar en cambiar de entidad bancaria" [IBM,2013], ello quiere decir que frente a errores como mal funcionamiento del sistema que utilizan que tienen cargos financieros inadecuados (por ejemplo : cobrar una comisión donde no se

debería), banca por internet (transacciones en línea) que no se ejecutan correctamente o no están disponibles el 50% de los clientes de una entidad bancaria optarían por cambiar de banco.

Figura 1.1.-Tolerancia del cliente [IBM, 2013]

Conviene mencionar que no existe una ley o regulación específica en Perú que indique que las entidades bancarias deben contar con un área que certifique las aplicaciones desarrolladas. Sin embargo la Ley 26702 banca y seguros [CRP, 2012] indica que se deben mantener registros de las transacciones financieras por 10 años, para la realización de posibles investigaciones. Asimismo, se indica que deben desarrollar y ejecutar programas, normas, procedimientos y controles internos para prevenir y detectar delitos; esto se refiere a que a un sistema informático no se le debe permitir funcionar si este facilita la comisión de un delito las transferencias bancarias fantasmas que en realidad se tratan de transferencias dudosas. Además la Constitución Política del Perú [CRP, 1993] en su artículo segundo numeral 6, remarca el derecho a la privacidad sus datos informáticos, esto va de la mano con la seguridad y buen funcionamiento que el sistema tenga.

De otro lado es conocido que la mayoría de sistemas bancarios informáticos en el mundo utilizan productos que tienen muchos años operando en el mercado y que los bancos lo adquieren y lo adecuan a su propia organización. Estos sistemas informáticos suelen ser grandes y se encargan de casi toda las operaciones de los bancos y deben ser actualizados de manera continua sea por necesidades de adaptación al marco regulatorio o necesidades propias de nuevas funcionalidades del banco o por detección de defectos.

Considerando que algunos bancos en nuestro país tiene una realidad parecida sobre la realidad de sus sistemas informáticos principales y sus procesos, se denominará **EL BANCO** al caso que se presenta y sobre el que se desarrolla todo el proyecto.

En EL BANCO se creó hace algunos años un área de Certificación de software orientado principalmente a las pruebas funcionales de los sistemas de información críticos utilizando únicamente los entornos semejantes a los utilizados para el desarrollo del producto y realizando todo el proceso de pruebas de manera manual.

Esta área de Certificación se constituyó en un punto clave en la calidad de los sistemas informáticos críticos de EL BANCO pero también se convirtió en cuello de botella para toda el área de mantenimiento de los sistemas informáticos.

El área de Certificación, se encarga de realizar las pruebas a todos los sistemas informáticos que han sido objeto de modificaciones y además se encarga de la aprobación de los pases a producción, que también lo realiza teniendo en consideración el resultado de las pruebas.

En el área de Certificación, se realizan dos tipos de atenciones, que según el origen se pueden denominar: proyectos e incidencias. Estos tipos de atención se explican a continuación:

- Proyectos: Se refieren a los aplicativos o componentes nuevos, los cuales se integran – la mayoría de las veces – a los sistemas informáticos críticos del banco. Los proyectos tienen un presupuesto específico para la realización de las pruebas (certificación) que suelen definirse en varios meses (alrededor de seis) y cuenta con una planificación con una cantidad de recursos establecidos.

- Incidencias: Se refieren a ocurrencias o defectos encontrados en el ambiente de producción (ambiente donde funcionan los sistemas informáticos que dan la cara al usuario) y que exponen un mal funcionamiento de los sistemas informáticos. Las modificaciones provenientes de las incidencias deben ser vueltas a probar para asegurar la corrección del incidente.

> Según IEEE, un incidente es cualquier ocurrencia de un suceso que requiere investigación [IEEE 1008 1989]

Las personas que trabajan en el área de Certificación encargadas de realizar las pruebas funcionales tienen un rol de "Analista-probador", perfil que les permite estimar, planificar y ejecutar las pruebas del componente de software que será atendido. Por lo general, una estrategia del área es que las solicitudes de atención que reciben, cuando se refieren a trabajos (componentes) relativamente grandes, son descompuestas en componentes más pequeños que serán asignados a un analista-probador para la gestión y ejecución de las pruebas del componente. Adicionalmente se organizan de modo tal que se definen equipos especializadas por sistemas informáticos críticos.

El área de Certificación cada vez que realiza una atención sigue, de manera general, las siguientes actividades:

- revisión de documentación,
- estimación del requerimiento,
- planificación del requerimiento,
- preparación de los casos de pruebas
- recolección de la data necesaria para las pruebas,
- ejecución de las pruebas,
- redacción del informe final, y
- pase a producción.

Cada uno de los pasos (elementos de la lista anterior) requiere de una aprobación, la que es otorgada por la persona responsable de la atención (proyecto o requerimiento) del área de Certificación.

La solicitud de atención es decir “petición de las pruebas”, en relación al manejo de recursos humanos, tiene una mecánica de funcionamiento como una factoría. Al generarse una atención, la misma, luego de pasar ciclos de gestión previos, como análisis, estimación y planificación, es atendida por un grupo de recursos que realizan las pruebas. La cantidad de recursos es dinámica, esto quiere decir que varía según la complejidad y tiempo estimado de las pruebas. El área de Certificación es transparente para las demás áreas por lo que el servicio puede ser visto como una caja negra, como si se tratase de una “máquina” de realización de pruebas. Un aspecto importante a remarcar es que el área de Certificación es un órgano ajeno al área de desarrollo por lo que se espera haya objetividad en la realización de las pruebas pues ningún recurso de Certificación ha formado parte del desarrollo (codificación) del componente a probar.

En el área de Certificación se tiene tres tipos de perfiles de profesionales. El primero es el probador en adelante “tester” (20 personas), los analistas (cinco personas) y el líder del equipo (una persona). Considerando que lo usual es que la carga de trabajo excede la capacidad del equipo actual, el esquema de trabajo dentro del Área es por asignación de prioridades a cada uno de las peticiones de pruebas. Las prioridades van a indicar qué petición se va a atender antes que otra y además queda implícito que mientras la petición sea más importante va a necesitar – no necesariamente siempre – más recursos para su ejecución.

Como el personal constituye un recurso escaso, éste no siempre está disponible para su asignación a las “peticiones de pruebas”. En repetidas ocasiones se ha presentado el caso que el recurso tiene que ser reasignado a otra petición de pruebas que sobrepasa en prioridad al que están atendiendo. Dentro de este esquema de asignación y re-asignación se tiene presente el perfil y conocimientos del tester y la petición de prueba priorizada para ser asignada.

Adicional a lo antes descrito se puede mencionar lo siguiente:

- Reasignación de recursos: debido a los cambios de prioridad en atención se debería poder cambiar los recursos que atienden determinada tarea sin problema. (Esta necesidad de gestión queda como un trabajo futuro).

- Re ejecución de pruebas: Al ejecutar un caso de prueba y encontrar un error (por ejemplo: una manera incorrecta de funcionamiento del componente probado) es necesario, una vez corregido el error, re- ejecutar nuevamente la prueba para así verificar la correctitud del componente.

- Pruebas de regresión: Ya que los sistemas críticos del banco son constantemente modificados es necesario probar los componentes que interactúan con el componente cambiado para asegurar su correcto funcionamiento, lo que conlleva a realizar las mismas pruebas reiteradas veces por cambio.

El sistema informático que se propone como proyecto de fin de carrera pretende cumplir con lo siguiente:

- Registrar casos de prueba: Al trabajar sobre un sistema de información crítico se permite crear casos de prueba que se pueden parametrizar. De esta manera la lógica necesaria es grabada por el tester pero los datos (data de entrada) podría cambiar en cada ejecución.

- Crear un conjunto de casos de prueba: Lo que permite agrupar los casos de prueba previamente creados, reutilizando los mismos de manera que solo haga falta cambiar los datos de entrada.

- Ejecutar casos de prueba: luego de tener el “conjunto de casos de prueba” ejecutarlo, de manera que se pueda obtener un resultado según lo que se está probando.

- Generador de reportes: Necesario para presentar los resultados de la ejecución así como un resumen de casos correctos e incorrectos. Con este reporte se puede determinar si el resultado de la prueba cumple con los criterios de salida.

El proyecto de fin de carrera busca crear una infraestructura de pruebas capaz de dar soporte a la automatización “pruebas de sistema” de regresión, ejecución de casos de prueba, gestión de casos de prueba y ejecución de pruebas de regresión orientada a

sistemas de larga vida.

1.3.- Objetivo general

Desarrollar una infraestructura de software para realizar pruebas automatizadas para grandes sistemas de información que están en continuo mantenimiento en el contexto de sistemas de información bancario

1.4.- Objetivos específicos

Los siguientes son los objetivos específicos, los cuales son necesarios cumplir para lograr el objetivo general

- OE1: Caracterizar una infraestructura de software para la gestión y realización de pruebas automatizadas.
- OE2: Definir una arquitectura que se integre de una forma no invasiva al producto y permita una gestión automatizada de distintos casos de prueba.
- OE3: Construir con tecnología adecuada al sistema de información al cual se tiene que integrar en el piloto.
- OE4: Realizar un piloto de la infraestructura de pruebas.

1.5.- Resultados esperados

Los resultados establecidos para el proyecto son:

- Informe de las características que debe cumplir la infraestructura de software para la gestión realización de pruebas (OE1).
- Prototipo de software que administre y ejecute los casos y conjunto de pruebas almacenados (OE2).
- Documento de estructura de almacenamiento de los casos de prueba y conjunto de casos de prueba (OE2)

- Documento de arquitectura de software (OE3).
- Informe de los resultados del piloto realizado (OE4).

1.6.- Métodos, metodologías y procedimientos

Para el desarrollo del proyecto se ha establecido utilizar el NTP-ISO/IEC TR 29110-5-1-2 [NTP-2012]. Las razones son las siguientes:

- El proyecto cuenta con un número limitado de recursos humanos (una sola persona) lo cual dificulta el adoptar algunos modelos o metodologías existentes.
- La NTP mencionada está orientada a pequeñas organizaciones, incluye un proceso de gestión del proyecto y el proceso de desarrollo de software (ver Figura 1.5). Por lo que la gestión de proyecto y producto está, alineados entre sí.
- La NTP define con mayor detalle las actividades que se tienen que realizar.

La NTP-ISO/IEC 29110-5-1-2 es una traducción (adaptación) de la norma internacional ISO/IEC 29110-5-1-2 [NTP, 2012]. La Norma recoge las buenas prácticas necesarias para una pequeña organización que desarrolla software. Además es de libre descarga en el portal de INDECOPI o la ISO.

En lo que sigue de este documento se hará referencia a la NTP-ISO/IEC 29110-5-1-2 como “VSE Basic Profile” o simplemente “VSE BP” que es la denominación usual en Internet.

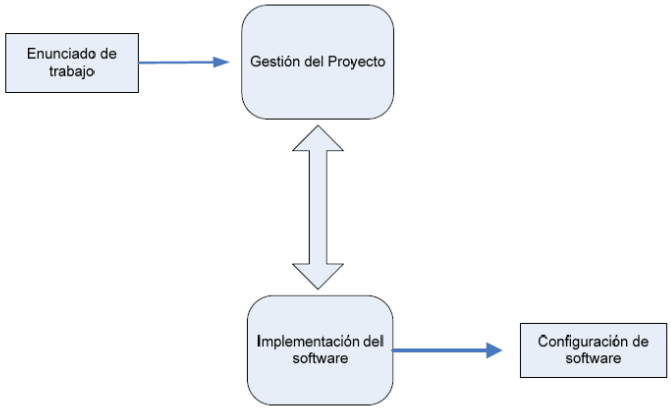

Figura 1.2-Procesos de la guía del perfil básico [NTP, 2012]

1.7.- VSE BP aplicado a gestión de proyecto

Para la gestión de proyecto se utilizarán las tareas referidas en el proceso de gestión de proyectos de la VSE-BP y que se presentan a continuación:

- Ejecución del plan de proyecto: Se tendrá como entrada el enunciado de trabajo y de acuerdo a ello se realiza el plan de proyecto. Finalmente es aprobado por el que haces las veces de cliente (osea el asesor). Nótese que el enunciado de trabajo cumple el OE1.

- Avance del proyecto monitoreado contra el plan de proyecto: Para un continuo control, se consideran controles de avance por parte de los profesores del curso en oportunidades con fechas definidas a lo largo del curso (ocho veces en tesis 1 y tres veces en tesis 2) y en otras coordinaciones con el asesor.

- Solicitudes de cambio: Esto es primordialmente durante las reuniones con el asesor. En caso se identifique algunos ajustes y sea conveniente cambiar ciertos aspectos del proyecto de fin de carrera, se tendría que evaluar y medir el impacto para luego cambiarlo. Esto quedaría registrado en un acta de reunión.

- Reuniones de revisión con el equipo de trabajo y el cliente: El cliente en este caso será asumido por el asesor y no hay equipos pues es solo un desarrollador

(el tesista) sostendrán reuniones de trabajo. Se llevará una bitácora de reuniones y modificaciones.

- Proceso de gestión de riesgos: El análisis en este punto será el mínimo y razonable para ello se realizara una matriz de riesgos para tener mapeados posibles amenazas y métodos para contrarrestarlos en un futuro. En la Tabla 1 se presenta la primera versión del plan de riesgo.

- Control de versiones de software: Es imprescindible tener un control de versiones de software. Por dos motivos: el primero es debido a cualquier falla humana sin dolo; el segundo motivo para mantener un registro con evidencia del avance del proyecto. Además de contar con una copia de respaldo (backup). Se usará un repositorio local y un repositorio en la nube (googlecode) ayudados con el cliente tortoise. De esta manera se tendrá siempre un doble resguardo de las versiones.

Descripción	Factor de riesgo	Impacto	Acción
Falta de acceso a información	Bajo	Alto	Se cuentan con los stakeholders correspondientes.
Alcance que sobrepase los recursos disponibles.	Bajo	Alto	Se tienen continuas reuniones con el asesor a fin de perfilar correctamente el proyecto de fin de carrera y delimitar correctamente el alcance.
Cambio de tema de proyecto de fin de carrera	Bajo	Alto	Reajustar los cronogramas trazados colocando días de trabajo los fines de semana.

Realizar las labores planificadas con retraso	Medio	Medio	Reajustar los cronogramas realizados. De ser necesario tomar días no laborables.

Tabla 1 - Riesgos relevantes al inicio del proyecto

La lista anterior es agrupada en cuatro actividades:

- Planificación del Proyecto
- Ejecución del plan de Proyecto
- Evaluación y Control del Proyecto
- Cierre del Proyecto

Las actividades definidas cumplen con lo establecido en la VSE – BP

Finalmente para graficar la metodología que se utilizó para la gestión del proyecto se puede observar la Figura 1.3.

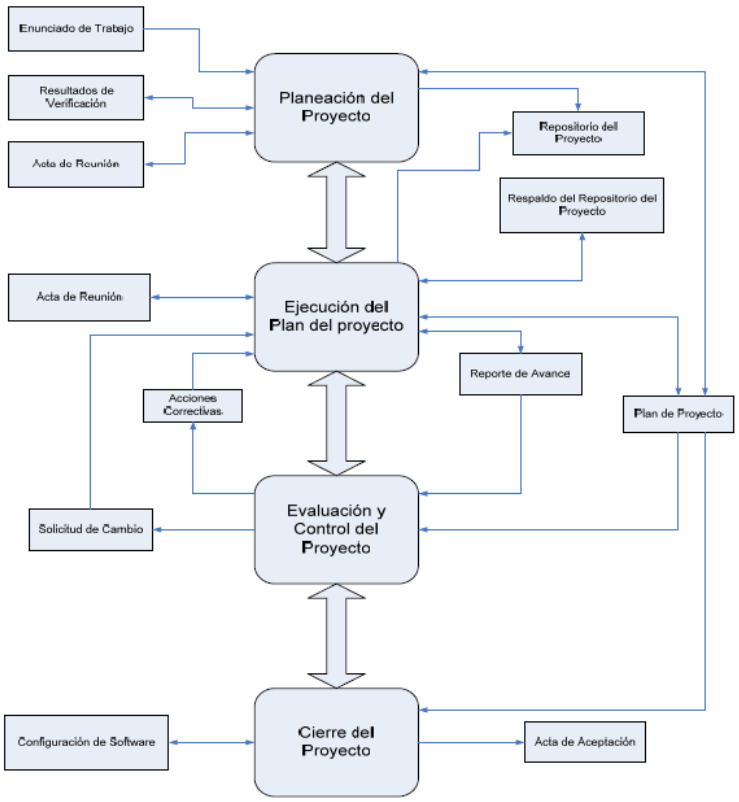

Figura 1.3.-Diagrama del proceso de Gestión del Proyecto [NTP-VSE]

1.8.- VSE-BP en la elaboración del producto

Se consideran las siguientes actividades de la elaboración de producto:

- Inicio de la Implementación de software
- Análisis de requisitos de software
- Arquitectura y diseño detallado del software
- Construcción del software
- Integración y pruebas del software
- Entrega del producto.

Las actividades mencionadas son las elementales y se recomienda no reducirlas ya que el VSE es de por sí una reducción de estándares más complejos y orientados a grandes organizaciones.

1.9.- Alcance y limitaciones

En esta sección se presenta el alcance y limitaciones del proyecto esto con el objetivo de tener claro las dimensiones del mismo.

Alcance

El presente proyecto de fin de carrera está contextualizado en el ámbito bancario y enfocado en la parte de Certificación (mediante pruebas) de los sistemas informáticos que han sido modificados. Este proyecto es importante para el grupo de bancos considerados por: la importancia que tiene la confianza que debe tener el cliente sobre el banco y por la obligación hacia el cliente de brindarle seguridad a sus datos. Se tiene acceso, por el ámbito laboral, a expertos en sistemas informáticos que están en continuo mantenimiento los mismos – sistemas informáticos- que no serán reemplazados en muchos años.

Esta realidad de sistemas grandes que tienen años de vida y que seguirán operando es muy frecuente en la mayoría de bancos en nuestro país.

Limitaciones

En esta sección se colocan los puntos que limitan el desarrollo de la infraestructura de pruebas de software.

- Falta de disponibilidad: Debido a que EL BANCO maneja información privada y delicada de clientes reales se ha previsto desarrollar un sistema bancario a menor escala pero con las mismas características. De esta manera será posible probar la infraestructura de pruebas en un ambiente lo muy parecido al real.

- Información confidencial no accesible: Ya que es una empresa bancaria, está restringido el acceso a información que comprometa la seguridad de sus

sistemas. El acceso a información brindada presenta dicha restricción, frente a esto mencionar que se ha recabado la información necesaria vital para poder desarrollar el producto y se tiene acceso – mediante entrevistas- a personal que trabaja en dicha institución.

1.10.- Viabilidad y justificación del proyecto

A continuación se presenta el análisis de viabilidad del proyecto y razones para la realización del mismo.

Viabilidad

Se han considerado criterios para el análisis de viabilidad: Viabilidad económica y Análisis de necesidades

- Viabilidad económica

Se usarán recursos de costo asumible por el tesista o de costo cero. Entre ellos se puede enumerar lo siguiente:

- ✓ Recursos metodológicos
- ✓ Recursos teóricos
- ✓ Mano de obra directa
- ✓ Tecnología para el desarrollo
- ✓ Asesoramiento de especialistas
- ✓ Análisis de necesidades

El proyecto tiene como objetivo desarrollar un producto que contribuya a mejorar la situación del área de TI del Banco

Justificación del proyecto

Se puede mencionar que:

- Es conveniente el desarrollo de un sistema de información que de soporte a la automatización y gestión de pruebas con el fin de reducir los costos y agilizar el proceso de pruebas, para este tipo de empresas, orientado a pruebas de regresión.

- Es necesario evaluar las necesidades en el proceso de construcción de software del negocio y el impacto que tiene sobre el producto de software con el objetivo de determinar cuantitativamente los riesgos que se toman al desarrollar un producto sin un proceso correcto de pruebas y con ello observar los riesgos que se mitigan y los que se eliminan al utilizar la sistema de información propuesto.

- Es conveniente un sistema de pruebas de manera que contribuye a la cultura de calidad y manejo de buenas prácticas dentro de proyectos de software. Además incentiva el conocimiento colectivo y la afluencia de nuevas ideas para afinar detalles de la aplicación, el proceso de

- construcción de software y la afluencia de ideas para posteriores desarrollos sobre el sistema a desarrollar.

1.11.- Plan de actividades

Se presenta una lista de actividades inicial. La misma ha sido modificada a lo largo del proyecto (Tabla 2).

#	Descripción	F. Inic	F. Fin
1	Elección y justificación del tópico	27/08/2012	10/09/2012

2	Problemática contextualizada del tema de tesis. Marco teórico. Estado del Arte del problema a resolver	24/09/2012 24/09/2012	30/09/2012 30/09/2012
3	Redacción del objetivo general del proyecto de tesis	01/10/2012	27/08/2012
4	Presentación de los objetivos específicos (mínimo	08/10/2012	16/10/2012
5	Resultados esperados del proyecto de tesis	29/10/2012	02/11/2012
6	Presentación del problema, objetivogeneral corregido y objetivos específicos corregidos (mínimo 4) y resultados esperados.	03/11/2012	10/11/2012
7	Alcance y limitaciones del proyecto de tesis. Metodología de proyecto y de producto a seguir	11/11/2012	18/11/2012
8	Planificación temporal del proyecto de tesis. Justificativa del proyecto del proyecto de tesis. Justificativa de la solución propuesta al problema en el proyecto de fin de carrera. Correcciones completas	19/11/2012	25/11/2012
9	Sustentaciones	26/11/2012	26/11/2012
10	Documentación formulación de requisitos de sw	06/01/2013	30/01/2013
11	desarrollo de sw	04/05/2013	19/06/2013

12	corrección de documentación	08/06/2013	19/06/2013

Tabla 2 - Plan de actividades

CAPÍTULO 2: Marco Conceptual

En esta sección se presenta los conceptos necesarios para comprender las definiciones utilizadas en el presente proyecto de fin de carrera así como una revisión de estado del arte y las herramientas utilizadas que tienen particularidades parecidas al presente proyecto.

2.1. Conceptos

- *Calidad:* La calidad según la ISO 9000:2005 [ISO 9000,2005] es “el grado en el que un conjunto de características inherentes cumple con los requisitos”. Los requisitos pueden ser explícitos o implícitos. La calidad es por hoy un aspecto importante de un producto por lo que debe ser considerada y asegurada de manera eficaz y eficiente.

  Si bien la calidad le da valor agregado al producto, los costos relacionados al aseguramiento de la misma deberían mantenerse dentro de ciertos márgenes. Por ello, para evitar costos excesivos se considera tener un manejo de calidad tanto en el producto como en el proceso; pues de esta manera se pueden detectar errores o “desviaciones” en etapas tempranas cuando el costo de corregirlo es relativamente bajo, comparado si se encuentra el mismo error en la etapa final del desarrollo del proyecto [Harrington , 1990] .

- Proceso de pruebas: Para asegurar un correcto desarrollo de un proceso de pruebas es conveniente tener una base sobre la cual trabajar. Se ha previsto trabajar con ISTQB [Graham, 2008]; un estándar de-facto del mercado de elaboración inglesa y basada en la norma IEEE829 [IEE829,

  2007]. ISTQB ofrece un enriquecimiento de la norma de la IEEE colocando estrategias determinadas y listas desarrolladas para su aplicación inmediata.

  En el modelo de buenas prácticas para el desarrollo de pruebas se indica que el proceso de pruebas debe llevarse desde la incepción del proyecto siguiendo un camino paralelo al desarrollo, de esa manera se asegura un seguimiento y

proceso de control de la calidad continuo en todo el proceso de desarrollo de software.

El modelo que es utilizado por la metodología ISTQB es el modelo en W (Ver Figura 2.1) que es un modelo que acompaña a todo el desarrollo del proyecto de software desde requerimientos hasta las pruebas de aceptación y que indica que actividades se deberían realizar en cada una de las etapas[Graham, 2008].

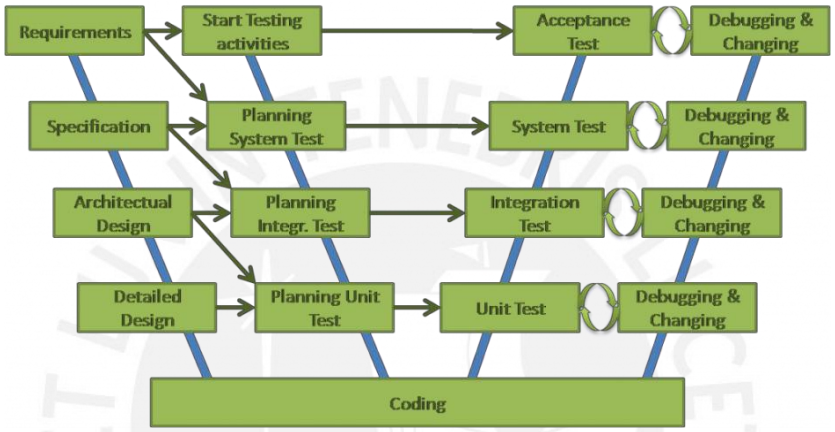

Figura 2.1-Modelo W [Ewald Roodenrijs, 2012]

El objetivo del proceso de pruebas es asegurar que la implementación ha sido llevada de manera correcta, de tal modo el software está listo para su entrega. Para el desarrollo de pruebas es recomendable tener determinados puntos a considerar de otro modo el proceso podría ser desordenado y no se cumpliría el objetivo del proceso de pruebas; además como la mayoría de todo proyectos cuenta con recursos limitados (por ejemplo el tiempo, dinero, personal).

Para un adecuado proceso de pruebas es necesario considerar los siguientes puntos:

- El alcance de las pruebas: circunscribe lo que se va probar.
- El criterio de salida de las pruebas: indica cuando una prueba se da por finalizada.
- La conformidad de las pruebas: Es la validación de las pruebas.
- La documentación de las pruebas: Es la evidencia del proceso de pruebas.

2.2. Pruebas

Las pruebas tienen objetivos puntuales. Según los objetivos se determinan que tipos de pruebas aplican. Para cada escenario en particular, además de contar con una estrategia de prueba. Resulta difícil probar todos los escenarios ya que las entradas y las circunstancias en las que se puede encontrar el componente

probado son muchas y en una situación real, se cuenta con recursos limitados por lo que es necesario priorizar las pruebas necesarias e ir escalando con las prioridades hasta consumir los recursos disponibles. En resumidas cuentas, es conveniente priorizar los aspectos a revisar en un proceso de pruebas, priorizando lo esencial [Graham, 2008].

2.3. Enfoques y Técnicas de pruebas

Los tipos de enfoques o estrategias para casos de pruebas que se tiene son lo siguientes:

- Pruebas por clases de equivalencia: estas consisten en agrupar datos de entrada en grupos de similar comportamiento y de tal manera que se puedan reducir la cantidad de casos a construir, asumiendo que todos los valores incluidos en esos grupos tienen similar comportamiento lo que tiene como consecuencia la reducción en la cantidad de casos a un número limitado factible de probar. Hay que recalcar que el punto de realizar pruebas no es hacer uso exhaustivo de todos los datos y explotar todos los escenarios, es más; es recomendable el usar casos de prueba que, con el menor de los esfuerzos, abarquen más de las opciones a probar de tal manera que se pueda "utilizar el avance de los demás casos de prueba" para tener un avance para el siguiente caso de prueba a ejecutar es recomendable [Román, 2007].

- Pruebas basadas en experiencia: estas pruebas toman como punto de partida la experiencia del tester. Muchas veces es necesario probar extensivamente en un tiempo reducido. De tal manera es necesario priorizar las pruebas. Factores como la experiencia, conocimientos y necesidades de la empresa influyen en la priorización. ISTQB menciona, como uno de los paradigmas en su metodología, que: "usualmente todos los errores suelen estar agrupados" [Graham,2008], de tal

manera que en estos escenarios donde el tiempo apremia es útil un acercamiento a la ejecución de pruebas de esta manera, personas que suelen estar relacionadas con la empresa pueden priorizar las pruebas en determinada sección considerando que en ese lugar puede encontrarse la mayoría de errores críticos.

- Análisis de valor límite: Consiste en analizar el comportamiento del sistema informático a probar, utilizando como datos de entrada valores situados en el límite de una partición de equivalencia. Por ejemplo: se tiene un campo “Monto del retiro”, solo aceptará valores de tipo numéricos, de esta manera se pueden identificar tres intervalos de valores: valores negativos hasta el 0 (<0) y valores de cero al límite del retiro (X>) y del límite del retiro a más (>X). Se tiene el cero y “X” como él limitante de intervalos, por lo tanto conviene analizar el comportamiento del sistema utilizando el valor cero (0 y X) .[Graham, 2008]

2.4. Tipos de pruebas

- Prueba de caja blanca: En la prueba de caja blanca es donde se puede observar el proceso de transformación de “la entrada” en “la salida”, ello quiere decir que se tiene acceso al código y se puede encontrar un fallo, defecto y error a la vez. Este está más inclinado a personas directamente relacionados al código – entiéndase desarrolladores- ; dentro de las pruebas de caja blanca se tiene principalmente las siguientes [Román, 2007] :

  - ✓ Pruebas de cobertura.
  - ✓ Complejidad ciclomática.

- Pruebas de caja negra o funcionales: Basan su lógica en que “la entrada” se transforma en “la salida” y el proceso que lo hace es transparente para el usuario, de tal manera que el objetivo de esta prueba es determinar si el software está trabajando de manera correcta (verificación)[Beizer, 1995].

  Dentro de las pruebas de caja negra se tienen las siguientes:

  - ✓ Prueba de humo: Suelen hacerse para comprobar rápidamente si el software funciona de manera correcta, estas contienen un número limitado de casos de prueba. También se utilizan para descartar

rápidamente si el ambiente está disponible para la realización de las pruebas [Beizer, 1995].

- ✓ Pruebas funcionales: Son realizadas –como su nombre indica – para comprobar que el software funciona de acuerdo a la documentación con ello se refiere a que el software verificado.

- ✓ Pruebas de Aceptación: Validación por parte del cliente (usuario final) que el sistema cumple con los requisitos de software.

- ✓ Pruebas de regresión: Se denomina pruebas de regresión a probar componentes que no han sido alterados sin embargo, corre el riesgo que su comportamiento no sea el habitual ya que están relacionados con un componente que si ha sido modificado. De tal manera, las pruebas de regresión se realizan a posibles componentes que pueden haber sido afectados de manera colateral al modificar un elemento con el cual interactúan [Graham, 2008].

Además de las pruebas mencionadas existen otras como pruebas de stress, de penetración, carga y demás que escapan del alcance del proyecto.

Automatizar pruebas conlleva a un riesgo alto y una inversión alta, es muy común ver proyectos de automatización fallidos [Christer, 2004]. Por lo que es conveniente tener cierto criterio para automatizar y saber qué hacer.

Automatizar consiste en que, ciertos procesos o procedimientos se ejecutan de forma no asistida ello con el objetivo de agilizar el proceso teniendo en cuenta si el retorno de la inversión (ROI) es positivo, ello implica que de nada sirve automatizar si el coste de la automatización es mucho mayor que el valor agregado que brinda a la empresa [Graham, 2008].

Para un proyecto de software es vital encontrar la mayor cantidad de defectos lo antes posible en nuestro caso en particular, encontrarlas antes del pase a producción ya que ello representa un costo menor al cliente.

CAPÍTULO 3: Estado del Arte

En esta sección se presenta información acerca de la resolución del problema antes planteado no se busca que lo resuelva de manera exacta sin embargo tiene que aproximarse a la solución.

3.1.-IBM RATIONAL QUALITY MANAGER

IBM Rational Quality Manager [IBM RQM, 2012]: impulsa la colaboración y la productividad a través del ciclo de vida de la calidad, permite a los equipos compartir la información sin problemas, usar la automatización para acelerar los calendarios de proyectos, y realizar informes sobre las métricas del proyecto para tomar decisiones bien fundamentadas; algunas de sus características son:

- Alineado con los cinco imperativos de Aplicación Lifecycle Management (planificación integrada, la trazabilidad de los artefactos relacionados, desarrollo de la inteligencia, la automatización y la colaboración, la mejora continua del proceso)
- Administra de manera proactiva el riesgo dando prioridad a las características y funciones que van a analizarse en función de su importancia y la probabilidad y el impacto.
- Mejor soporte distribuido geográficamente en el contexto de la comunicación del equipo con las características tales como avisos de eventos, chat integrado y trazabilidad automática
- Acelerar las pruebas manuales usando “Richtext”, “in-line images” y entrada de datos y validación asistida para la definición detallada de pruebas y ejecución
- Prueba de laboratorio para gestión y seguimiento, programar y ejecutar las pruebas de laboratorio sobre los activos físicos y virtuales

La Figura 3.1 muestra una captura de la pantalla de bienvenida al ingresar al RQM, muestra dos partes : la que está completamente a la izquierda que es una barra de trabajo que es desde donde va permitir explorar toda la aplicación y los datos que están grabados dentro de la aplicación. El resto de la pantalla es un lienzo que va mostrar el resultado de las opciones que estemos buscando que puede ser la creación de un plan de prueba o un caso de prueba entre otros.

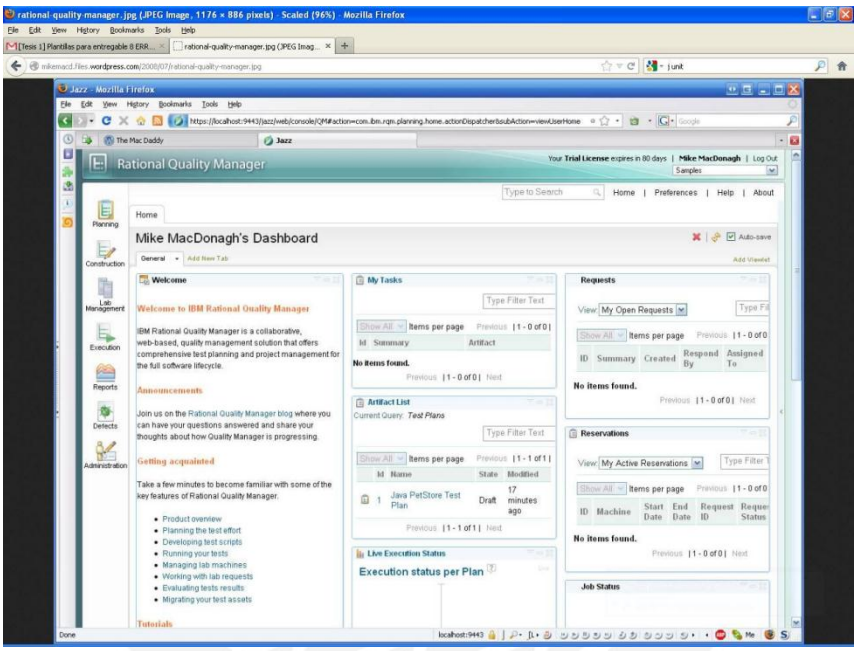

Figura 3.1 IBM RQM [IBM RQM, 2012]

3.2.-HP Quality Center

El software HP Quality Center (anteriormente HP Test Director) impulsa una eficaz y eficiente manera en el proceso global de la pruebas de aplicaciones y es compatible con los altos niveles de comunicación. HP Quality Center cuenta con varios módulos de gestión de requisitos, la liberación y Gestión del ciclo, plan de pruebas, laboratorio de pruebas, defectos, Gestión, y el tablero de informes, que se integran a la perfección para permitir la circulación fluida de información a través de etapas de prueba [HP QC, 2012].

HP Quality Center maneja una estructura organizada en proyectos esto quiere decir que se subdivide en proyectos y la información de casos de prueba, suites de pruebas y demás se almacenan en un repositorio central de manera que todos los usuarios puedan tener acceso a la información. Debido a que HP Quality Center está basado en web, todos los usuarios pueden acceder a información crítica del proyecto independientemente de los límites geográficos y organizativos.

El HP Quality Center tiene la capacidad de dar soporte desde etapas tempranas del proyecto y no solamente desde la etapa de pruebas ya que es un software que también da soporte al Aseguramiento de la calidad (QA).

Los beneficios de HP Quality Center Enterprise son [HP QC, 2012].:

- Los analistas de negocio pueden definir la aplicación que se está probando, los requisitos y objetivos de las pruebas, basados en las prioridades del negocio.
- Los gerentes de control de calidad puede priorizar los esfuerzos de las pruebas basadas en riesgos del negocio.
- “Thread” de prueba (entiéndase que un thread de prueba es ejecutar una transacción a la vez en la aplicación y multithread que se ejecutan varias transacciones concurrentes) además de diseñar planes de prueba y desarrollar casos de prueba.
- Los ingenieros de automatización de pruebas pueden almacenar la automatización de las secuencias (HPQC permite grabar los flujos que se recorren en una prueba para luego utilizarlos como casos automatizados) de comandos y los activos en el repositorio de HP Quality Center.
- Los testers de control de calidad pueden ejecutar las pruebas manuales y automatizadas además de informar sobre los resultados y los defectos. Todo ello es soportado por la herramienta.
- Los analistas de negocio y los testers de control de calidad pueden gestionar varias versiones de las pruebas y de los activos de prueba, mientras que mantienen la integridad de los datos, ello quiere decir que el HPQM no permite modificar los datos fuera de la ejecución de la prueba.

- Los desarrolladores pueden revisar y corregir los defectos registrados en la base de datos de HP Quality Center.

- Los gerentes de proyecto pueden revisar los índices de calidad y decidir si una aplicación está lista para el lanzamiento.

- Los administradores de control de calidad pueden hacer cumplir los procesos y mejores prácticas a través de proyectos.

- Los analistas de negocios, desarrolladores y testers de control de calidad pueden compartir y reutilizar los activos de las bibliotecas a través de proyectos para reducir la duplicidad de esfuerzos.

- Los planificadores y los administradores de la versión pueden agregar métricas de calidad a través de proyectos.

La Figura 3.2 muestra el panel principal del HP Quality Manager, se puede ver en la parte izquierda la barra de módulos que muestra las distintas opciones desde las pruebas (que es lo que interesa) como el módulo de defectos, componentes del programa, entre otros. Se pueden ver varios módulos ya que se le puede añadir al HPQC múltiples funcionalidades extras a través de “addons” (extensiones), en la parte del lienzo central izquierdo se puede ver un árbol con varios temas de interés, al desglosarlo se puede ver el detalle en la parte derecha del lienzo.

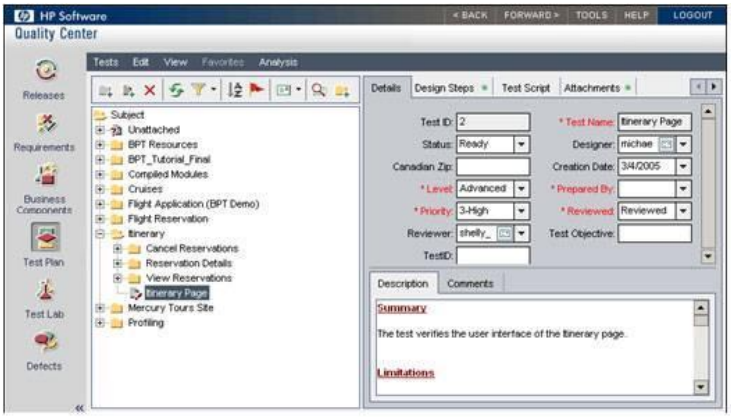

Figura 3.2.-HP- QC [HP QC, 2012]

3.3.-SELENIUM

"Selenium" es una extensión del navegador de internet "Mozilla Firefox". Este es un software de automatización de procesos o tareas para el navegador de internet, basa su funcionamiento en dos características [Selenium, 2012]:

- Automatización mediante la captura de las acciones: Ello indica que el aplicativo permite grabar la acción que realiza el usuario en contacto con el navegar y luego reproducirla, pudiendo parametrizar los datos de ingreso.

- Programación con el lenguaje nativo de selenium: es también posible programar todas las acciones que realiza la aplicación, no siendo necesaria el grabado de "macros".

Finalmente si bien es un software amigable para automatizar, tiene ciertas desventajas:

- Solo se pueden automatizar aplicaciones basadas en exploradores de internet y los cuales son ejecutados en el Mozilla ya que la extensión de selenium solo funciona para dicho explorador de internet.

- El lenguaje que utiliza para la programación de los casos no es intuitivo.

- Selenium es un software que tiene solo una jerarquía de dos niveles: casos de prueba y suite de prueba (la cual agrupa casos) no hay manera de poderlos diferenciar, de este modo se puede decir que es una herramienta que ha desarrollado más la parte de ejecución que la de gestión por ello es que las suites y los casos de prueba se manejan de manera manual y se clasifican de manera manual (por carpetas) ya que la herramienta no provee de ayuda para la tarea mencionada.

Sin embargo hay que resaltar que es un software gratuito.

3.4.- GXTEST

Gxtest es una herramienta de automatización de pruebas funcionales específica para GeneXus. Se resalta que ambas herramientas pertenecen a la compañía “Abstracta”. Conviene mencionar que GeneXus es un entorno de programación que genera aplicaciones en distintas arquitecturas. GxTest se basa en la misma. La principal ventaja que se puede rescatar es que al estar completamente integrado el GeneXus, se basa en los metadatos por lo que no depende de la arquitectura donde finalmente es compilada la aplicación. Sin embargo su gran desventaja es que no funciona con ninguna otra aplicación o entorno que no sea recibiendo soporte o interactuando con GeneXus [Gtest, 2012].

3.5.- TOSCA

Tosca, se centra en la ejecución automatizada de pruebas funcionales y de pruebas de regresión de software tanto de software GUI y sin GUI. Cuenta con herramientas que permite que realice funciones como administrar, crear, ejecutar e intercambiar. Tiene gran integración con distintos lenguajes de programación, aplicaciones en host, exploradores web y distintos hardware y protocolos.

La principal ventaja yace en que es compatible con software de terceros. Además de contar con hardware y protocolos. Tosca tiene como principal ventaja el ser tan amigable, capaz y permitir a un usuario no experto el poder crear casos de prueba automatizados y que puedan ser leídos por personal técnico. Su principal desventaja es que solo pueden ser usados por organizaciones medianas y grandes debido a su costo [Tosca, 2012].

3.6 JUNIT

El Junit es un framework para escribir y ejecutar pruebas automatizadas para aplicaciones en java. Dentro de sus ventajas se tiene que permite insertar aserciones (muy parecido al selenium). Ello permite comparar un valor con otro por medio de una condición y el resultado es verificado y de ello depende el resultado de la prueba (un grupo de aserciones da un caso de prueba y un grupo de casos a una

suite). Se puede agrupar los casos en suites para una mejor organización. Una desventaja es la carencia de una interfaz completamente gráfica ya que la única manera de automatizar es codificando además de ser únicamente orientado a pruebas unitarias [Junit, 2012].

3.7 EasyMock

EasyMock es un framework para generar mockobjects de interfaces en tiempo de ejecución. Los mockobjects son stubs y drivers usados para realizar las pruebas. Los Stubs y Drivers ayudan a controlar la alimentación de datos o comportamiento de un componente con el que se relaciona el componente a probar. Como desventaja tiene que no es gráfico y que solo se integra con java. Se puede resaltar que es de uso gratuito [EasyMock, 2012].

3.8 Cobol Unit

El Cobol Unit nace como ayuda para los desarrolladores de cobol ya que ellos no contaban con ninguna herramienta para la automatización de pruebas además buscaba solucionar un gran problema : "probar un programa o método sin necesidad de modificar el código" , este problema se presentaba ya que el Cobol, acrónimo de Common Business-Oriented Language no venía de la mano de ninguna metodología de desarrollo, tampoco de documentación ni código comentado de tal manera que el código era de difícil mantenimiento y entendimiento.

Cobol Unit es un framework que ayuda, sin necesidad de saber el funcionamiento del programa que se prueba, a probar el funcionamiento del mismo teniendo en cuenta solo dos cosas los datos de entrada y el resultado. Es necesario codificar y compilar un programa ayudado del Cobol Unit luego de eso solo basta con ejecutarlo.

Tiene como ventajas el ser gratuito, ser un framework que ayuda a probar código en Cobol como desventaja tiene la rigidez para adaptarlo a otro tipo de pruebas como unitarias o de sistemas [Vaujour, 2013].

3.9.-Conclusiones sobre el estado del arte

La Tabla 3 en la que muestra una comparación de productos que constituyen en alguna medida una posible solución para el problema planteado y que se analiza junto a otras soluciones existentes en el medio local descritas en la sección anterior. Las columnas detallan la solución mientras que las filas las características que se están comparando.

Herramienta	Ejecución y gestión de casos de pruebas	Costo	Orientación
Producto a realizar	Ejecuta, gestiona y administra.	La solución de manera inicial solo consideraría costos de mano de obra directa.	Orientado a sistemas de información de larga vida.
Rational quality manager	No ejecuta casos	Se adquiere no solamente el producto sino el servicio de Pruebas por lo que el producto viene acompañado de la solución referente al proceso de pruebas	No tiene una orientación específica ya que es indistinto del sistema al cual va orientado.
HP Quality Center	No ejecuta casos	Se adquiere.	No tiene una orientación específica ya que es indistinto del sistema al cual va orientado.
EasyMock	No presenta interfaz de gestión	Gratis	Aplicaciones en java
JUNIT	No presenta interfaz de gestión	Gratis	Aplicaciones en java
TOSCA	Completo soporte a la creación, ejecución, gestión y especificación de casos de prueba así como la introducción de datos manuales	Se adquiere.	general

GXTEST	Si cuenta con soporte a gestión para pruebas de regresión, aceptación, integración y sistema.	Es una herramienta que necesita otra que es la base para su funcionamiento, ambas se adquieren por separado	Orientado a todo tipo de software desarrollando en GeneXus
SELENIUM	Mínimo de gestión, forma muy primitiva de agrupar casos de prueba. Permite ejecutar un set de casos así como cada caso	Es gratuito. Es una herramienta adicional en el explorador de Internet Firefox	Orientado a aplicaciones web
COBOL UNIT	Mínimo, la gestión	Es un framework gratuito, herramienta que Prueba código cobol sin modificar la fuente	Orientado a pruebas funcionales en cobol y ejecución completa con sentencias en cobol o CICS

Tabla 3 - Comparación Soluciones

Al revisar las herramientas se puede decir que en todos los casos carecen de ciertas particularidades inherentes al sistema donde se utilice. Esto se debe a que cada sistema tiene adaptaciones y comportamiento muy particular debido tanto al giro del negocio como a políticas y restricciones que puedan haber adoptado al desarrollar su sistema informático.

El proyecto plantea una alternativa de solución que contiene dos partes: gestión y automatización de casos de prueba. La gestión comprende el manejo de estimación, diseño, planificación, ejecución y manejo de casos de prueba. Por otro lado la automatización se encarga de ejecutar de manera no asistida los casos de prueba a través de scripts de prueba. La solución rescata las características positivas de las herramientas utilizadas y añadiendo un valor agregado propio del escenario tomado como problemática (ámbito bancario). La infraestructura se integra de forma no invasiva a la aplicación que va probar.

CAPÍTULO 4: Análisis

En esta sección se van a detallar los requerimientos del proyecto de fin de carrera además de la especificación de casos de uso más significativos del sistema. Se define de manera clara y precisa las funcionalidades y la manera en las que estas se desarrollaran.

4.1.-Identificación de requerimientos

Esta sección se presentará los requisitos del sistema informático a realizar, de esta manera se tendrá una visión general de las funcionalidades del sistema. Se dividirán en requerimientos funcionales y no funcionales. Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema y lo no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema [Sommerville, 2005].

Las tablas presentadas a continuación están conformadas por un número correlativo, la descripción y la prioridad. La descripción de la prioridad se puede ver en la Tabla 4.

- **Requerimientos Funcionales**

No.	Descripción	Prioridad
1	El sistema permitirá crear, modificar y eliminar casos de prueba	1
2	El sistema permitirá crear, modificar y eliminar sets de pruebas	1
3	El sistema permitirá asociar casos de prueba con sets de pruebas	1
4	El sistema permitirá ejecutar casos de pruebas en cobol	1
5	El sistema permitirá ejecutar sets de pruebas	1

6	El sistema permitirá crear, modificar y eliminar perfiles de acceso al sistema	1
7	El sistema permitirá crear, modificar y eliminar flujogramas que representen el sistema probado	4
8	El sistema permitirá crear sets de prueba a partir de la selección de partes de los flujogramas.	4
9	El sistema permitirá versionar los casos de prueba	1
10	El sistema permitirá versionar las sets de prueba	1
11	El sistema permitirá elaborar reportes de resultados de pruebas	1
12	El sistema permitirá crear, modificar y eliminar indicadores de pruebas, según los parámetros disponibles	4
13	El sistema permitirá exportar e importar casos de prueba	2
14	El sistema permitirá exportar e importar sets de prueba	2
15	El sistema permitirá planificar un proceso de pruebas	2
16	El sistema permitirá estimar un ciclo de pruebas	2
17	El sistema permitirá aprobar un resultado de pruebas	2
18	El sistema permitirá reportar y asignar errores a usuarios	3
19	El sistema permitirá crear, modificar y eliminar solicitudes de atención (requerimientos de software)	1

20	El sistema permitirá reasignar solicitudes de atención en base a la priorización	1

Tabla 4 - Requerimientos funcionales

- **Requerimientos no Funcionales**

No.	Descripción	Prioridad
1	El sistema estará elaborado usando el lenguaje java y en el IDE eclipse	1
2	el sistema operativo cliente será en Ubuntu	1
3	El manejador de base de datos será MS MYSQL	1
4	El software se entregará en un cd autoinstalable	1
5	El sistema permitirá a los usuarios acceder con credenciales	1
6	Para el almacenamiento de Test Cases se utilizará el “test case description language 2.0”	1

Tabla 5 - Requerimientos no funcionales

Número	Descripción
1	Alta
2	Media
3	Baja
4	deseable

Tabla 6.- Clasificación de prioridades

4.2.-Análisis de la solución

En esta sección se detallara de manera abstracta la solución y la estructura del sistema informático a desarrollar.

- **Análisis del requerimiento**

Esta sección está orientada a la especificación de los requerimientos más significativos del proyecto

  - Catálogo de actores

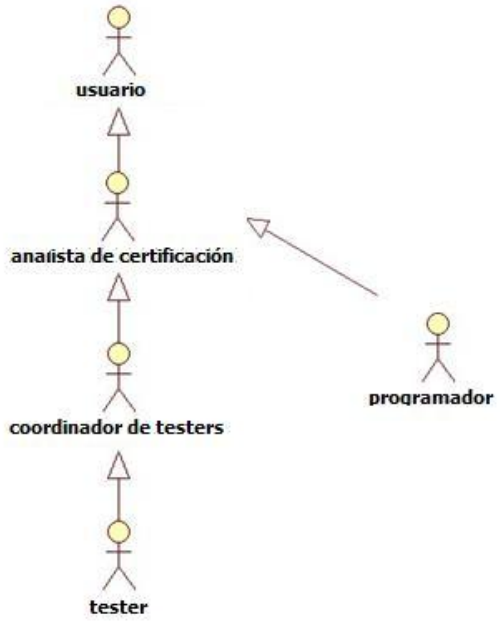

Figura 4.1.- Catálogo de actores

La Figura 4.1 muestra el nivel de jerarquía entre actores.

- Usuario: es el usuario final, aquel que valida si las pruebas son conformes.
- Las pruebas son conformes cuando cumplen con las especificaciones del sistema y cumplen el alcance dispuesto por el usuario.

- Analista de certificación: es la persona que aprueba las pruebas a realizarse. Tiene conocimientos del negocio y es el primer filtro frente a los errores reportados.

- Coordinador de testers: es el encargado de gestionar a los testers, distribuir sus tareas y asignar las tareas de probar.

- Tester: es el recurso que va realizar las pruebas y reportar los errores.

- Programador: es el recurso al que finalmente se le van a derivar los errores, él se encargará de corregir el error y versionar el código corregido.

4.3 .-Casos de uso por paquete

En esta sección se mencionan los paquetes y funcionalidades vistas de una manera abstracta (a manera de descripción) más representativos del sistema informático desarrollado así como la interacción entre dichas funcionalidades.

- Paquete de pruebas

Este paquete contiene los casos de uso asociados a la sección de pruebas

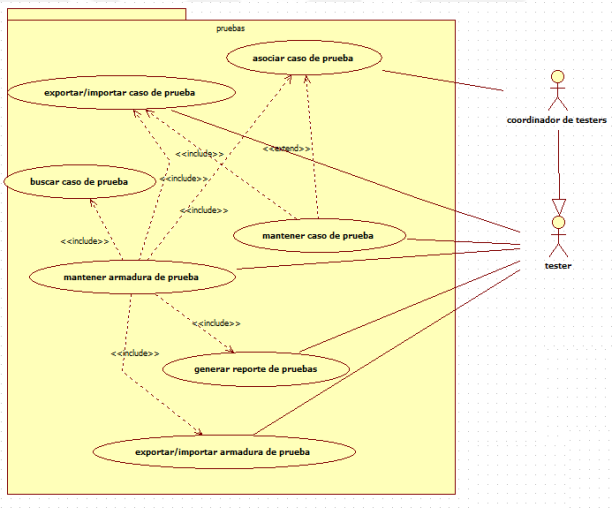

Figura 4.2.-Diagrama de paquete de prueba

- Asociar caso de prueba: Permite asociar un caso de prueba a un set de prueba.

- Exportar/importar caso de prueba: Permite exportar e importar casos de prueba. Esto permite poder migrar el contenido completo (o librería) de

- caso de prueba de una plataforma a otra. Si el caso fuera creado de manera local o si fuese necesario migrarlo debido a versiones esta funcionalidad lo permitiría de la misma manera la importación.

- Buscar caso de prueba: Esta función permite realizar una búsqueda por filtros y por etiquetas (tags) de los casos de prueba. Esta función se utiliza a lo largo de los mantenimientos de casos de prueba así como la asociación de casos de prueba a sets de pruebas.

- Mantener caso de prueba: Esta función permite, agregar, modificar y eliminar casos de prueba. Dentro de la fase de agregar se adjuntara el script de prueba –que posteriormente será ejecutado-, descripción, nombre, etiquetas de búsqueda rápida entre otros.

- Mantener set de prueba: Permite agregar, modificar y eliminar un set de pruebas que es una agrupación de casos de pruebas. Estos suelen tener dos tipos: los agrupados por funcionalidad. Por ejemplo: distintos casos de retiro) o por prueba a realizarse (por ejemplo: casos relacionados a ITF (mencionado en la problemática) que pueden pertenecer a secciones de cajero, plataforma, web, entre otros.

- Generar reporte de pruebas: Permite generar el reporte de pruebas el cual consolida los resultados de pruebas.

- Exportar / importar set de pruebas: Permite exportar e importar sets de prueba. Esto permite poder migrar agrupaciones de casos de prueba.

- Paquete de gestión

Este paquete contiene los casos de uso asociados a la sección de gestión del proceso de pruebas.

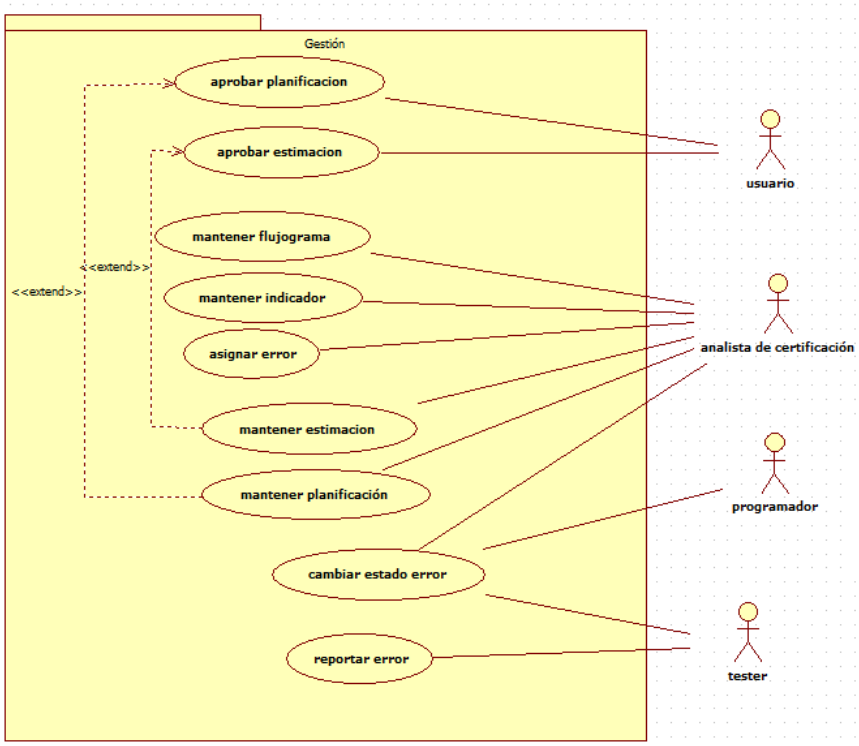

Figura 4.3.– Diagrama paquete de gestión

- Mantenimiento de estimación: Permite agregar, modificar y eliminar una estimación la cual consiste en medir de manera previa los recursos a utilizar en el proceso de pruebas.

- Mantenimiento de planificación: Permite agregar, modificar y eliminar una planificación la cual consiste en disponer una fecha indicada para el inicio de pruebas así como los recursos específicos a utilizar.

- Mantenimiento de indicador: Permite agregar, modificar y eliminar indicadores previamente parametrizados en base a la información disponible. Los cuales permiten generar un reporte que mide el estado del indicador.

- Reporte de error: Permite reportar un error. Si el caso ejecutado no presenta un resultado satisfactorio es necesario corregir el funcionamiento del componente probado es por ello que el error es reportado al recurso responsable para su solución.

- Cambio de estado de error: Permite cambiar el estado del error. El cambio del estado del error va de la mano con la etapa del error que son reportado- cuando el error es informado-, respuesta de desarrollo – cuando el error ha sido corregido-, persistencia del error – cuando el error se vuelve a presentar pese a su corrección-, entre otros.

- Asignar error: Permite asignar un responsable del error, el cual se encarga de su solución y de esta manera proseguir con el flujo de trabajo.

- Paquete de Seguridad

Este paquete contiene los casos de uso asociados a la sección de seguridad

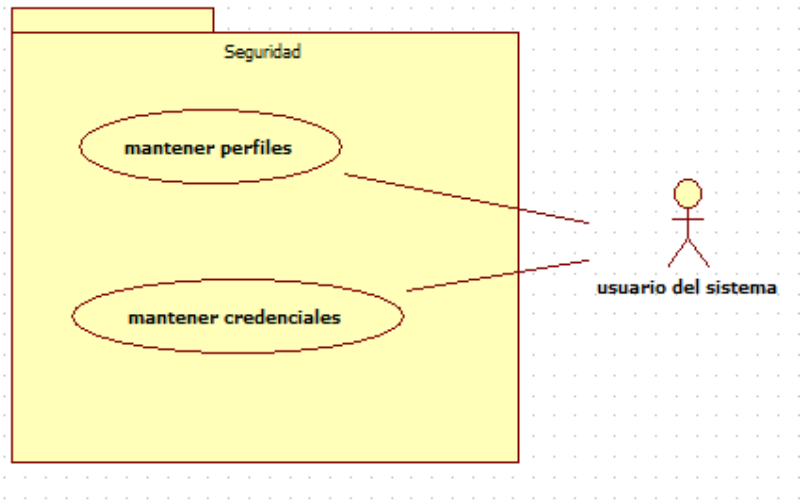

Figura 4.4-Diagrama paquete de seguridad

- Mantener perfiles: permite modificar los accesos a secciones del programa.

- Mantener credenciales: Permite la gestión del acceso a la arquitectura de pruebas.

4.4 Especificaciones de casos de uso

En esta sección se especifican los casos de uso más importantes del sistema

CÓDIGO	SRF001
CASO DE USO	Mantener caso de prueba
DESCRIPCIÓN	Este caso de uso permitirá registrar, modificar o eliminar un caso de preba.
ACTORES	Tester
PRE-CONDICIÓN	El tester debe tener los permisos necesarios para ejecutar la tarea. Se debe tener un script de cobol previamente.
FLUJO PRINCIPAL : crear caso de prueba	
1. El tester selecciona la opción “nuevo caso de prueba” 2. El sistema muestra el formulario de registro de nuevo caso de prueba (código, sistema, subsistema, transacción, nombre del caso, descripción caso, script cobol, resultado esperado, código set de prueba). 3. El tester ingresa todos los datos obligatorios (set de prueba no es un campo obligatorio). 4. El sistema registra el caso de prueba. 5. El tester coloca la opción guardar.	
POST-CONDICIÓN	

Se registra satisfactoriamente el caso de prueba.
FLUJO ALTERNATIVO: Modificar caso de prueba
1. El tester busca y selecciona el caso de prueba a modificar. 2. El sistema muestra el registro del caso de prueba seleccionadoprueba (código, sistema, subsistema, transacción, nombre del caso, descripción caso, script cobol, resultado esperado, código set de prueba). 3. El tester modifica los parámetros que desee y selecciona la opción aceptar.
POST-CONDICIÓN
Las modificaciones del caso de prueba seleccionado se lograron con éxito.
FLUJO ALTERNATIVO: Eliminar caso de prueba
PRE-CONDICIÓN: El caso de prueba tiene la misma estructura que admite el sistema
1. El tester busca y selecciona el registro a eliminar. 2. Se muestra una pantalla de confirmación 3. El usuario selecciona la opción que eliminar 4. El sistema elimina desactiva el caso de prueba indicado, de tal manera que
POST CONDICIÓN:
Se elimina el caso de prueba satisfactoriamente.

FLUJO ALTERNATIVO: Importar caso de prueba
1. El tester selecciona la opción importar caso de prueba. 2. El selecciona dentro del explorador de archivos el caso de prueba a importar. 3. Seleccionar el caso a importar y presionar el boton “importar”.
POST CONDICIÓN: El caso ha sido importado

CÓDIGO	SRF002
CASO DE USO	Mantener caso set de prueba
DESCRIPCIÓN	Este caso de uso permitirá registrar, modificar o eliminar un set de prueba.
ACTORES	Tester
PRE-CONDICIÓN	El tester debe tener los permisos necesarios para ejecutar la tarea.
FLUJO PRINCIPAL : crear set de prueba	

1. El tester selecciona la opción “nuevo set de prueba” 2. El sistema muestra el formulario de registro de nuevo set de prueba (código, sistema, subsistema, nombre de la set de prueba, descripción, casos de prueba, planificación, estimación). 3. El tester ingresa todos los datos obligatorios (subsistema, casos de prueba, planificación y estimación no son campos obligatorios). 4. El sistema registra la set de prueba. 5. Eltester coloca la opción guardar.
POST-CONDICIÓN
Se registra satisfactoriamente el set de prueba.
FLUJO ALTERNATIVO: Modificar set de prueba
1. El tester busca y selecciona el set de prueba a modificar. 2. El sistema muestra el registro del set de prueba seleccionado (sistema, subsistema, nombre de la set de prueba, descripción, casos de prueba, planificación, estimación). 3. El tester modifica los parámetros que desee y selecciona la opción aceptar.
POST-CONDICIÓN
Las modificaciones del set de prueba seleccionada se lograron con éxito.
FLUJO ALTERNATIVO: Eliminar set de pruebas

1. El tester busca y selecciona el registro a eliminar. 2. Se muestra una pantalla de confirmación 3. El usuario selecciona la opción que eliminar
POST CONDICIÓN:
Se elimina el set de prueba satisfactoriamente.
FLUJO ALTERNATIVO: exportar caso de prueba
1. El tester selecciona la opción exportar set de prueba. 2. El tester busca y selecciona el set de prueba a exportar 3. El tester selecciona los casos de prueba que van a ser contenidos dentro del set de prueba a exportar.
POST CONDICIÓN: el set de prueba ha sido exportada

CAPÍTULO 5: Diseño

En esta sección se describirá la manera en la que está distribuido y organizado el sistema informático desarrollado en el presente proyecto.

- Arquitectura de la solución

En esta sección se presenta de manera abstracta la estructura, interrelación de componentes de la arquitectura de pruebas.

- Vista lógica del sistema

La vista lógica del sistema permite visualizar a grandes rasgos la manera en la que va a estar organizado el código dentro del sistema. Se toma como referencia el patrón de arquitectura MVC (Modelo Vista Controlador).

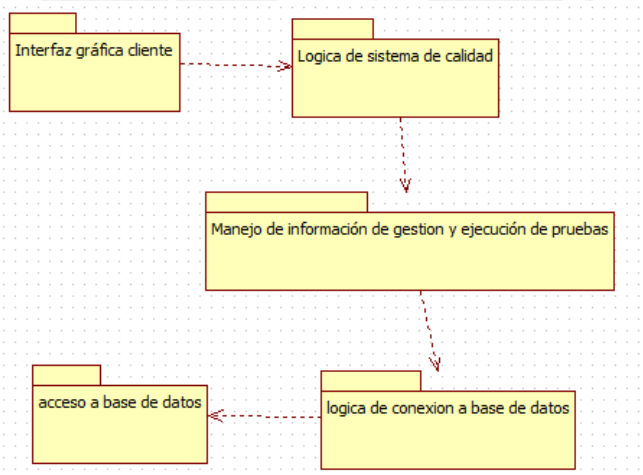

Figura 5.1.-Vista lógica del sistema

Se definen las siguientes cinco capas en la que va a estar distribuido el sistema:

- Interfaz gráfica cliente: Es la capa visual, es la capa que va interactuar directamente con los usuarios. En ella se puede encontrar pantallas y formularios. La interfaz necesita directamente de una lógica es por ello su dependencia con la capa “Bussinesslogic” (o lógica del negocio).

- Lógica del sistema de calidad: Es la capa del controlador, maneja toda la lógica del negocio y permite manejar los recursos ofrecidos por la capa de businessentity.

- Manejo de información de gestión y ejecución de pruebas: En esta capa se encuentra las clases del sistema como unidades como artefactos estáticos, no tienen ninguna lógica.

- Lógica de conexión a base de datos: Contiene las interfaces de conexión a la base de datos.

- Acceso a base de datos: La capa de Data Access contiene los accesos y credenciales para el ingreso a la base de datos.

- Vista de Despliegue

Esta vista muestra los nodos físicos en los que se distribuye la solución.

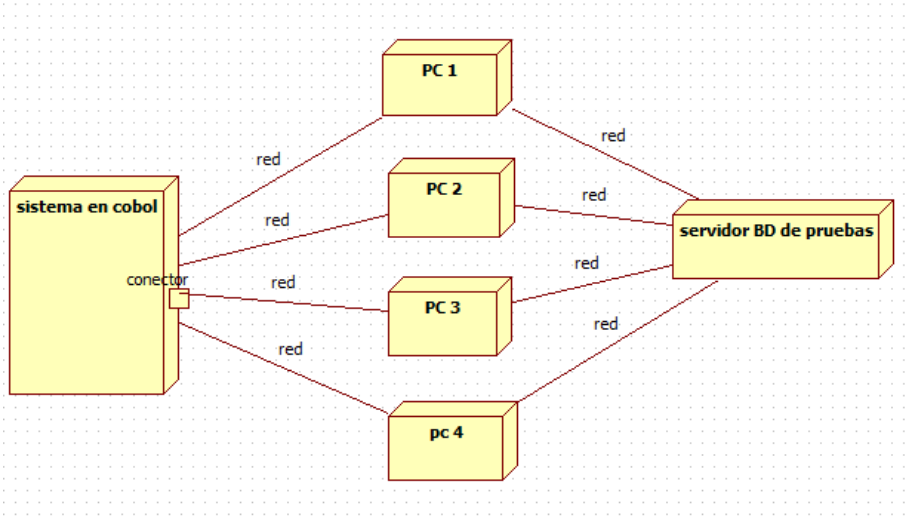

Figura 5.2.- Vista de despliegue

- PC: Computadoras personales de los usuarios que van a operar directamente con el sistema informático de pruebas.
- Servidor de BD de pruebas: Contiene la base de datos que va interactuar directamente con el sistema informático de pruebas.
- Sistema en Cobol: Es donde está alojado la aplicación o el grupo de aplicaciones a probar.

- Vista de Implementación

Esta sección describe los artefactos que se utilizan en conjunto para generar la solución. Todos ellos son necesarios para su distribución física, esta vista nos permitirá poder llevar versiones del sistema.

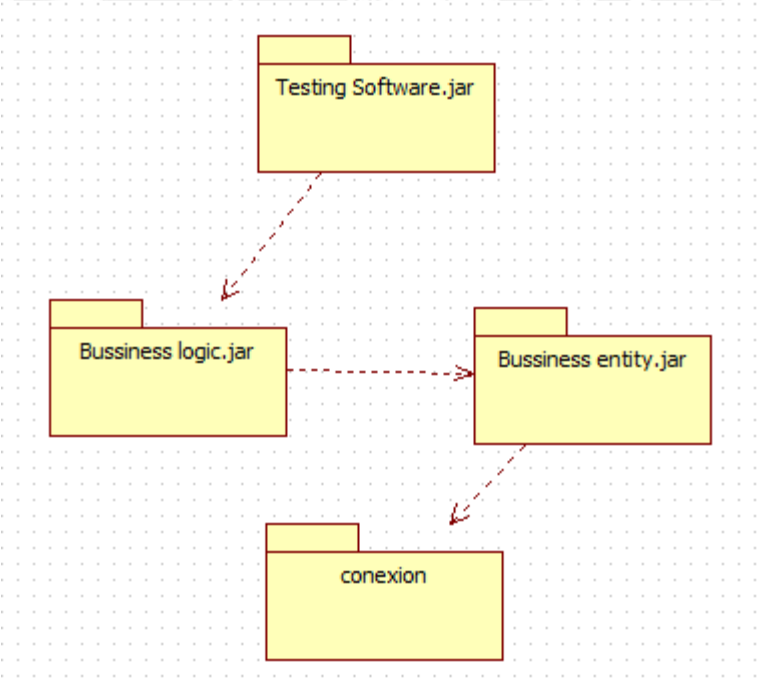

Figura 5.3.-Vista de implementación

- Ejecutable del sistema: Testing software.jar, es el ejecutable del sistema.

- Lógica del sistema : Componente que contiene la lógica del sistema
- (controlador) representado por “bussiness logic.jar”
- Entidades del sistema: Componente que contiene las entidades del
- sistema (Modelo) representado por “bussiness entity.jar”
- Componente de conexión: Componente que contiene todo lo necesario para establecer una conexión a la base de datos y al sistema a probar (conexión.jar).

Arquitectura del software

En esta sección se podrá ver a un muy alto nivel la estructura de la infraestructura realizada y la manera en que los componentes y las distintas capas se interrelacionan. Se puede ver en la Figura 5.4, cuatro secciones. La primera “sistema de información contiene toda la lógica del sistema, la segunda “middelware” contiene todas las conexiones tanto a base de datos como para interactuar con el sistema a probar, la tercera la “base de datos” es propiamente eso la base de datos donde se almacenan todos los datos relevantes del sistema y por último la cuarta “software a probar” simboliza el software el cual se va probar.

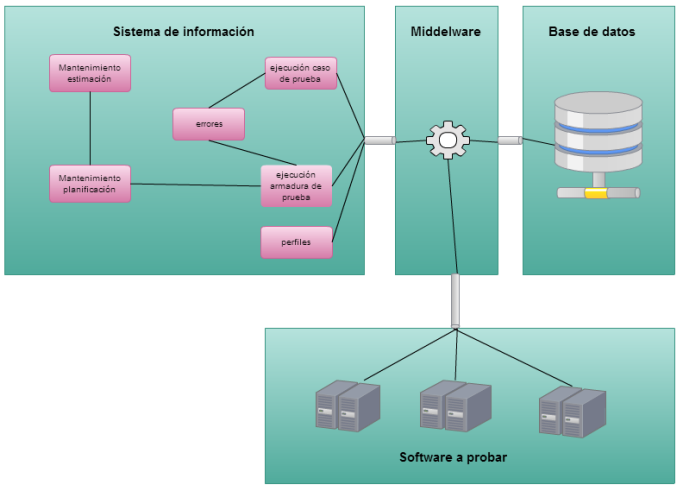

Figura 5.4.-Arquitectura del software

CAPÍTULO 6: Construcción

En esta sección se muestra la parte más significativa del sistema, colocando un flujo por las funcionalidades y partes del sistema informático desarrollado.

6.1 Almacenamiento de casos de prueba

Para almacenar los casos de prueba se utilizará el “test case description language 2.0”, en adelante TCDL, es un XML que describe casos de prueba y escenarios para los mismos.

La estructura de TCDL se plantea para un mejor ordenamiento y almacenamiento de los casos de prueba asimismo servirá para un mejor manejo de los casos de prueba y facilitará la inclusión de nuevas funcionalidades a futuro ya que la estructura lo permitirá.

El objetivo de utilizar el TCDL es introducir metadata para los casos de prueba que permitan facilitar la trazabilidad con información útil para el desarrollo de los casos. También permite la definición de escenarios de pruebas. El uso de este estándar es muy importante para desarrollos futuros que tengan como base el presente proyecto.

- Secciones de un archivo TCDL

Descripción de un caso de prueba: La descripción de caso de prueba, en un archivo TCDL, es la base de todo el fichero. La estructura (ver Figura 6.1) consiste en un identificador que sería el código, la versión del lenguaje y los namespaces.

```
1  id: RQM2013XX (obligatorio);
2  xml:(opcional);
3  dir: opcional);
4  xmlns="" (obligatorio);
5  xmlns:obligatorio);
6  xmlns:xlink="http://www.w3.org/1999/xlink": namespace for W3C XLink (obligatorio);
7  xmlns:html="": namespace for XHTML 1.x (obligatorio);
8  xmlns:xsd="";
9  xmlns:xsi="";
10 xsi:
```

Figura 6.1 .- Descripción de caso de prueba

La estructura del archivo TCDL (ver figura 1.2).

```
<?xml version="1.0" encoding="UTF-8"?>
<testCaseDescription id="RQM2013XX"
  xml:lang="es"
  xmlns=""
  xmlns:xsd=""
  xmlns:xsi=""
  xmlns:dc=""
  xmlns:xlink=""
  xmlns:html=""
  xsi:schemaLocation=""
  >
  <formalMetadata>
    <!-- child elements omitted -->
  </formalMetadata>
  <technologies>
    <!-- child elements omitted -->
  </technologies>
  <testCase complexity="atomic">
    <!-- child elements omitted -->
  </testCase>
  <rules>
    <!-- child elements omitted -->
  </rules>
  <namespaceMappings>
    <!-- child elements omitted -->
  </namespaceMappings>
</testCaseDescription>
```

Figura 6.2.- Estructura de archivo TCLD

6.2 Flujo de trabajo

El flujo de trabajo es una vista rápida de la manera en que el sistema informático elaborado va funcionar y cómo interactúan las distintas secciones del sistema.

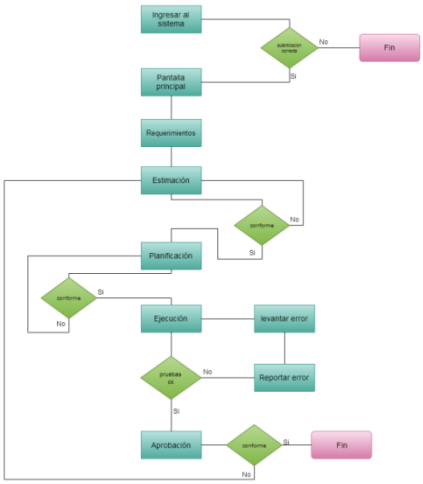

Figura 6.3.- Flujo del sistema informático

- Ingresar al sistema

  Al ingresar al sistema se colocan las credenciales: usuario y contraseña, luego se presiona el botón “Ingresar”, esto permite acceder al sistema. Si la autenticación no es correcta porque las credenciales no lo son, el sistema no permitirá acceso (ver Figura 6.4).

Figura 6.4.- Pantalla de autenticación

- Pantalla principal

Es en esta sección se presenta todos los accesos a las diferentes funcionalidades del sistema. Los accesos a las distintas funciones están bloqueados según los perfiles. La captura de pantalla muestra todas las opciones para mejor explicación (ver Figura 6.5).

Figura 6.5.- Pantalla principal

- Bandeja de tareas

En la bandeja de tareas se tiene un panorama principal de todos los requerimientos y sus estados. En ella se puede crear, modificar y eliminar nuevos requerimientos como también “ejecutar” que es pasar al siguiente estado del requerimiento (estimación, planificación, etc.) o devolver que consiste en regresar a la etapa anterior (ver Figura 6.6).

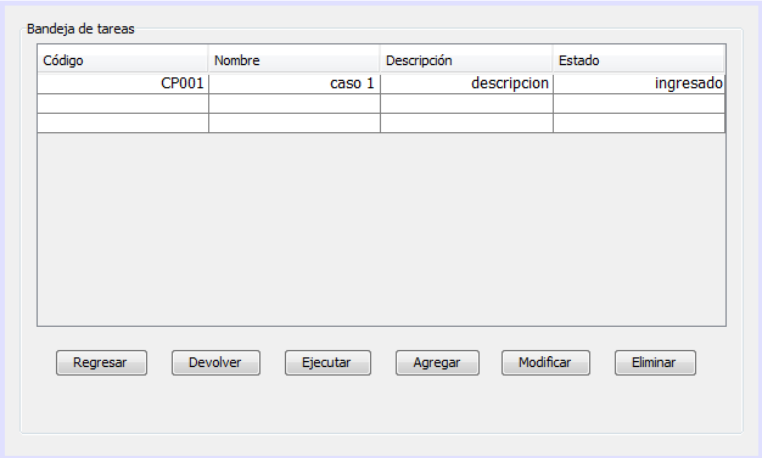

Figura 6.6.- Bandeja de tareas

- Requerimientos

Los requerimientos son las solicitudes de atención. En esta sección el analista de Certificación ingresa el requerimiento colocando información necesaria para su posterior análisis (documento de análisis detallado el cual contiene la descripción detallada del requerimiento y el documento de impacto el cual contiene los sistemas o subsistemas que está afectando el cambio) como se aprecia en la Figura 6.7.

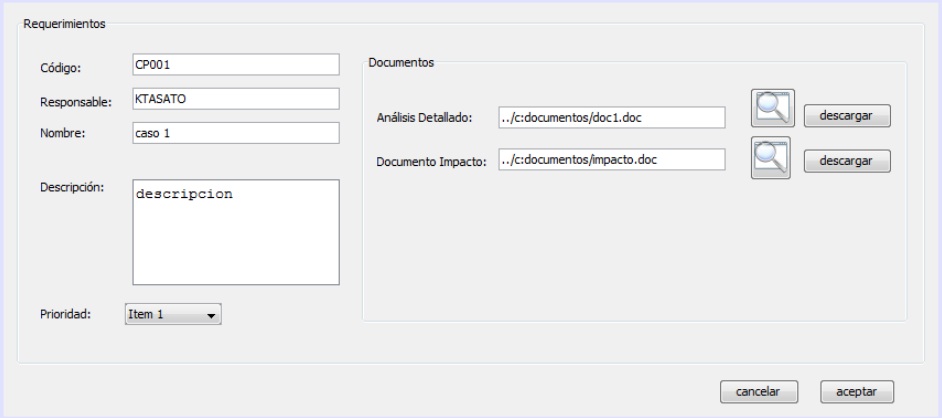

Figura 6.7.- Panel de ingreso de requerimientos

- Estimación

Es la medición preliminar de recursos que se necesitan para la atención del requerimiento. Se coloca las horas estimadas para atención, la cantidad de recursos y se detalla los tiempos utilizados para cada fase de la atención (análisis, diseño, ejecución y cierre). Se agrega el plan de pruebas el cual contiene las estrategias y el planteamiento específico de lo que se va probar como las funcionalidades que han sufrido cambios como las que no y es necesaria una prueba de regresión. Se coloca la descripción del caso a probar para que el usuario pueda identificar si se cubre el alcance. Conviene mencionar que en esta sección no se colocan los casos de pruebas solo la descripción de los mismos. La estimación llega a la bandeja de tareas del analista de Certificación, si es conforme pasa a la etapa de planificación, si no lo es se vuelve a estimar corrigiendo los cambios propuestos (ver Figura 6.8).

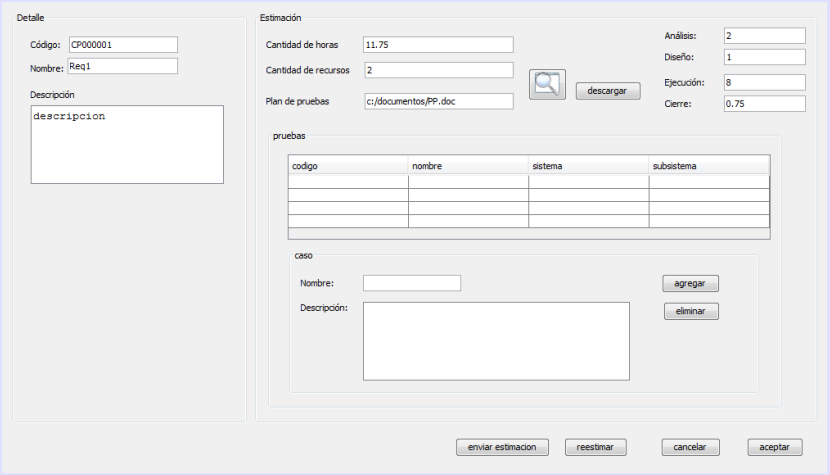

Figura 6.8.- Panel de ingreso de estimación

- Planificación

En la planificación solo se coloca la fecha de atención, esto va de la mano con la prioridad del requerimiento a atender (ver Figura 6.9).

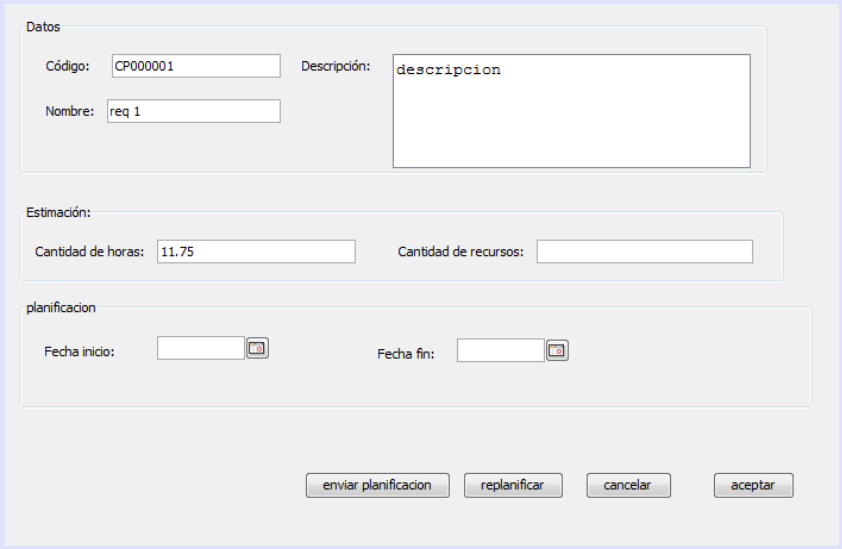

Figura 6.9.- Panel Ingreso de planificación

- Ejecución

El panel de ejecución contiene tres secciones: la asociación de casos, la ejecución de casos y el cierre. En la asociación de casos se selecciona cada uno de los registros presentados al usuario en la etapa de estimación y se le asocia un caso de prueba el cual contiene el script que va a ejecutarse y propiamente va probar la funcionalidad. Una vez completada la etapa de asociación de casos se marca la opción "completado asociación de casos" y eso nos permite ir a la pestaña ejecución de casos (ver Figura 6.10).

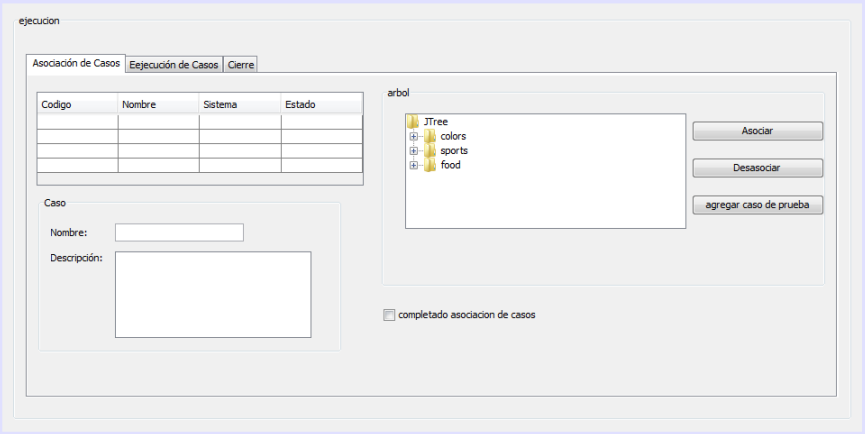

Figura 6.10.- Pestaña asociación de casos

En la ejecución de casos se selecciona registro por registro de la grilla, luego de ello se selecciona el botón "ejecutar", al ejecutar el caso mostrará el resultado en el cuadro de texto resultado: Pass, si el caso se ejecutó y tuvo un resultado exitoso, Fail si el caso tuvo algún error. Si el caso falla se activa la sección "reporte de errores" se selecciona la acción que en este caso sería "enviar error" y el usuario responsable que es el analista de desarrollo que recibe la observación para luego corregirla, se termina el proceso y se pulsa el botón "aceptar" con ello se envía la observación. En la grilla el caso de prueba se actualiza con el estado del resultado de la prueba, si se ha enviado un reporte de error se coloca el estado "obs" (observado) . Una vez que todos los casos tienen el estado "pass" se activa la pestaña de cierre (ver Figura 6.11).

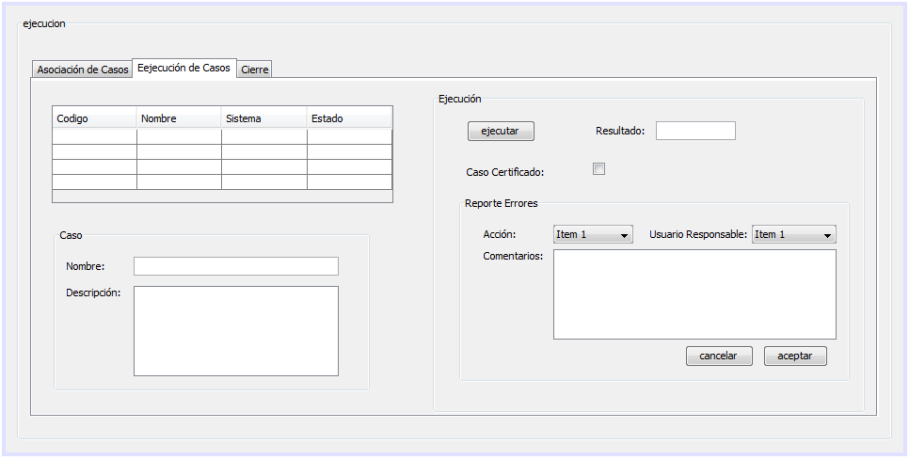

Figura 6.11.- Pestaña ejecución de casos

En el cierre se colocan comentarios u observaciones adicionales a la ejecución y se genera el reporte de ejecución el cual contiene el estado de la prueba así como la descripción de la misma. El reporte podrá ser descargado (ver Figura 6.12).

Figura 6.12.- Pestaña cierre de pruebas

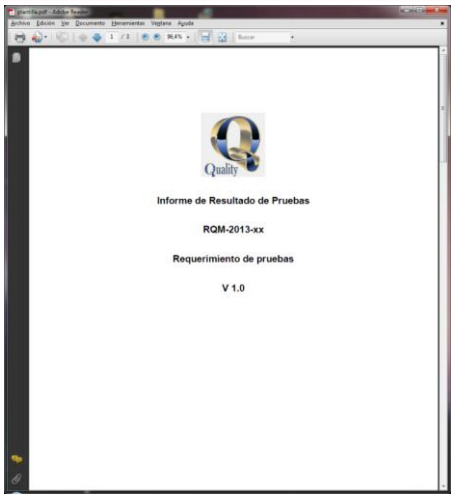

Figura 6.13.- Carátula de reporte de pruebas

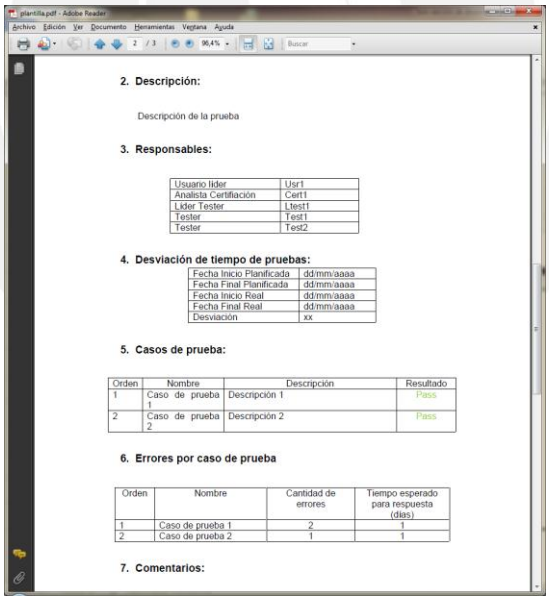

Figura 6.14.- Contenido de reporte de pruebas

- Aprobación

En la aprobación el usuario tiene acceso a detalle del requerimiento, casos de prueba y resultados para que pueda validar las pruebas ejecutadas. Si la aprobación es conforme el requerimiento se cierra y está en capacidad de pasar a producción. Si la aprobación no es conforme, por diversos motivos, el requerimiento regresa a la etapa de estimación para incluir lo adicional que indica el usuario (ver Figura 6.15).

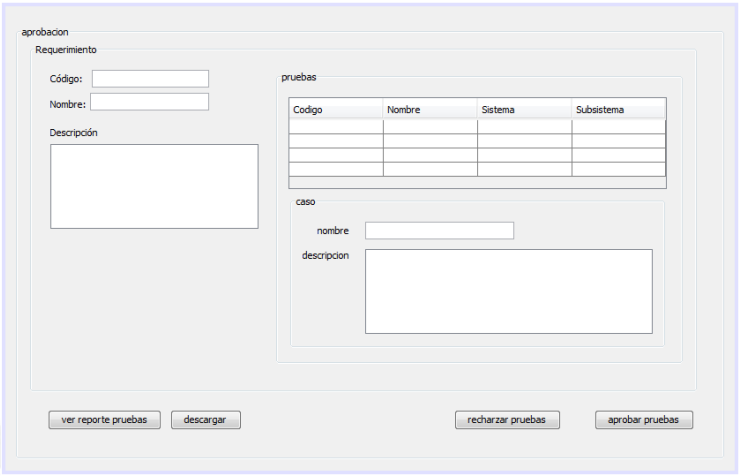

Figura 6.15.- Panel de aprobación usuario final

- Reporte de errores.

En el reporte de errores el recurso de desarrollo realiza una acción sobre la observación que puede ser: rechazarla, corregirla o colocar que es para un desarrollo posterior. Toda respuesta va acompañada de un comentario que detalla la acción, los errores se ven como una tarea en la bandeja de entrada. (ver Figura 6.16).

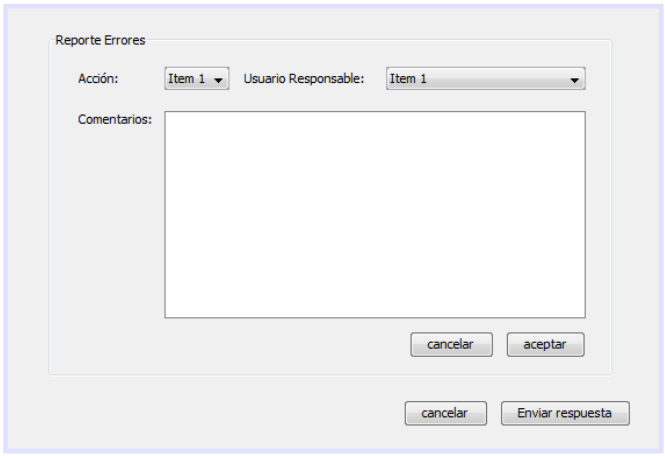

Figura 6.16.- Panel de reporte de errores

CAPÍTULO 7: Observaciones, conclusiones y recomendaciones

En esta sección se presenta las observaciones, conclusiones y recomendaciones.

7.1 Observaciones

- La ausencia en la definición del proceso de pruebas correcto impactó de manera negativa, pues con ello se requirió más tiempo para revisar este tema y así poder desarrollar un software

- Conviene un sistema de pruebas de manera que contribuye a la cultura de calidad y manejo de buenas prácticas dentro de proyectos de software. Además incentiva el conocimiento colectivo y la afluencia de nuevas ideas para afinar detalles de la aplicación, el proceso de construcción de software y la afluencia de ideas para posteriores desarrollos sobre el sistema a desarrollar.

7.2 Conclusiones

Al finalizar el proyecto se han llegado a las siguientes conclusiones:

- Se logró desarrollar una infraestructura de las pruebas de larga vida. Asimismo la infraestructura puede crear casos de pruebas y suites de prueba pues se utiliza TDCL el cual es un estándar de almacenamiento de casos de prueba.

- Se logró probar el funcionamiento correcto de la infraestructura de pruebas en un ambiente simulado (el software contra el cual se probó fue creado y consiste en un pequeño simulador de dos transacciones bancarias : retiro y depósito)

- Se logró desarrollar una infraestructura de pruebas de software que no modifica el código que prueba con ello se comprueba que es una solución no invasiva.

- Se logró desarrollar una infraestructura de pruebas que utiliza scripts en cobol el cual es la tecnología del programa objetivo que se está considerando a probar en el desarrollo.

7.3 Recomendaciones y trabajos futuros

En esta sección se exponen recomendaciones y mencionan trabajos futuros en base a la experiencia de desarrollo del proyecto.

Recomendaciones del uso y sobre el proceso de hacer la tesis:

- El sistema de información desarrollado en el presente proyecto no será eficaz si no va relacionado con una metodología de pruebas. Finalmente es solo una herramienta para la realización de una metodología.

- Es conveniente contar solo con el 70% del tiempo disponible para el desarrollo del producto ya que el otro 30% debería ser usado de contingencia en caso faltase tiempo para alguna sección del proyecto.

- Ya que es un sistema modular y por el mismo manejo de combinación de firmas, jerarquías y flujos de decisión es razonable pensar que puede convertirse en un sistema orientado a documentos (workflow).

- Añadiendo la capacidad de interactuar con otro tipo de interfaces (como las visuales directamente) se puede escalar el tipo de pruebas y realizar una prueba mucho más completa, una prueba funcional y funcional de toda una sección del sistema.

- Utilizando la sección que compara enunciados se puede generalizar y explotar convirtiendo un módulo del programa en un detector de cambios y un versionador de documentación o código automático, podría decirse una herramienta a fin de procesos de inspección.

- Se pueden enlazar con herramientas de pase de ambientes (herramientas que permiten la migración de objetos y componentes de un ambiente a otro, como del ambiente de desarrollo al ambiente de pruebas y luego al ambiente de producción) combinados con la función de flujograma podrían generarse automáticamente un diagnóstico de impacto al ver los módulos impactados con el pase y generar una versión inicial de set de pruebas básica a probar para garantizar que los componentes afectados funcionan de manera correcta.

Referencias bibliográficas

- [IBM, 2013] IBM ,Tolerancia a errores por parte de clientes frente a entidades bancarias. [En línea] [Citado el: 27 de marzo del 2013.] http://www.ibm.com/smarterplanet/es/es/banking_technology/ideas/index.ht ml

- [CRP, 2012] CONGRESO DE LA REPUBLICA DEL PERU,2012 ley 26702. Ley general de banca y seguros. 05 de Junio

- [CRP, 1993] CONGRESO DE LA REPUBLICA DEL PERU, 1993.Constitución Política del Perú.

- [IEE829 , 2007] Institute of Electrical and Electronics Engineers. 2007. Draft IEEE Standard for software and system test documentation. IEEE 829. New York : new york, 2007. 10016-5997.

- [ISO 12207, 2008] Institute of Electrical and Electronics Engineers. 2008, ISO 12207.ISO 12207-2008 segundaedicion 2008-02-01.

- [Harrington, 1990] Harrington, H. James 1990. Poor-Quality Cost.NewYork :Ediciones Diaz de Santos. 84-87189-58-X.

- [Graham, 2008]Graham, Dorothy. Foundations of software testing: ISTQB certification. london : Thomson Learning, 2008. Isbn – 987-1844809899.

- [Beizer, 1995] Beizer, Boris. Black-box Testing: Techniques for Functional testing on software and Systems. John Wiley & Sons,1995. Isbn – 978-0471120940.

- [Alarcón,2008] Ruth Alarcón, Carla Basurto, Abraham Davila. Infraestructura de pruebas para una plataforma de negocios: lecciones aprendidas de una experiencia académica. España: Asociación de Técnicos de Informática (ATI), 2008, Vol. 4. 1885-4486. 9 -19

- [Damm, 2005] Damm, Lars-Ola, results from introducing component -level test automation and test -driven development.. sweden :ElSEVIER, 2005, Vol. 79. 1001-1004.

- [Christer, 2004] Christer Persson, Nur Yilmazturk, Proceedings of the 19th IEEE international conference on Automated software engineering..washington :s.n., 2004. 0-7695-2131-2

- [Ewald Roodenrijs, 2012] Software testing and more. [En línea] [Citado el: 16 de 09 de 2012.] . http://www.testingthefuture.net/2009/09/the- w-model/.

- [IBM RQM, 2012] Vistazo rápido al producto. IBM. Racional quality manager. [En línea] [Citado el: 05 de 05 de 2012.] <http://www-01.ibm.com/software/rational/products/rqm/>.

- [HP QC, 2012] HP. Quality Center. [En línea] [Citado el: 05 de 05 de 2012.] <https://www.sqa.its.state.nc.us/library/pdf/HP%20Quality%20Center%20Overview.pdf>.

- [Selenium, 2012] blog. Selenium HQ.Selenium HQ. [En línea] [Citado el: 19 de 09 de 2012.]. www.seleniumhq.org.

- [Gtext, 2012] Gxtext, Pagina web. [En Linea] [Consultado el: 01 de 11 de 2012].http://www.genexus.com/productos/gxtest?es

- [Tosca, 2012] TOSCA, Pagina web. [En línea] [Consultado el:01 de 11 de 2012] http://www.tricentis.com/en/

- [Junit, 2012] Junit, Pagina web, [En Línea][Consultado el: 01 de 11 de 2012], http://www.junit.org/

- [Easy Mock,2012] EasyMock, Página web [En Línea][Consultado el: 01 de 11 de 2012], http://easymock.org/

- [ISO 9000,2005] Secretaria Central ISO en Ginebra. 2005, ISO 9000:2005. Sistemas de gestión de la calidad, 2005.

- [NTP-2012] Norma Técnica Peruana- Perfiles del ciclo de vida para las pequeñas organizaciones (PO).Parte 5-1-2: Guía de gestión e ingeniería: Grupo de perfil genérico. Perfil básico

- [Vaujour, 2013] Hervé Vaujour, Cobol Unit, Página web. [En

Línea][Consultado el: 29 de 03 de 2013]. https://sites.google.com/site/cobolunit/documentation/cobol-unit-architecture

- [Ramos, 2007] Técnicas cuantitavias las para gestión en ingeniería del softwareJavier tuya, Isabel ramos román, Javier dolado cosínIsbn : 987-84-9745-204-5

- [Sommerville, 2005] Ian Sommerville, ingeniería del software, sèptima edición. Pearson educaciòn, S.A. madrid 2005. isbn: 84-7829-074-5. ribera del loira,28. 28042 madrid españa

Anexos