Facultad 2

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Propuesta de Estrategia de Pruebas para los Software del Centro de Telemática

Autores: Alejandro Jorge Murillo Pérez

Alejandro Lorenzo Mojarena

Turora: Ing. Norbelis Leyva Montero

CoTutora: MSc. Violena Hernández Aguilar

Ciudad de la Habana, junio 2010

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo, y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo, en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Autores:

_____ _____

Alejandro Jorge Murillo Pérez Alejandro Lorenzo Mojarena

Tutor:

_____

Ing. Norbelis Leyva Montero

Datos de contacto

Alejandro Jorge Murillo Pérez,
Correo: ajmurillo@estudiantes.uci.cu
Ciudad de la Habana, Cuba

Alejandro Lorenzo Mojarena
Correo: amojarena@estudiantes.uci.cu
Ciudad de la Habana, Cuba

Ing. Norbelis Leyva Montero
Ingeniera en Ciencias Informáticas.
Correo: nleyva@uci.cu
Universidad de las Ciencias Informáticas
Ciudad de la Habana, Cuba.

DEDICATORIA

A toda mi familia, gracias por todo.

Murillo

A mi mamá, a mi papá y a mi bebé Robertico.

Alejandro

AGRADECIMIENTOS

A mi mamá por ser muy especial.

A mi papá por siempre estar presente para mí.

A Sergio por todo.

A mis abuelos por tanto cariño y amor y por malcriarme siempre.

A mi tía moni por quererme tanto.

A mi bb, por quererme tanto, ser tan valiente y volver a confiar en mí, por ayudarme en todo.

A mis tíos que siempre me han ayudado.

A Carlos y Nelson, mis hermanitos siempre.

A Norbelis, nuestra tutora, por soportarnos a pesar de tantos obstáculos en el camino.

Al equipo Metros de la 48 SNB gracias por dejarme ser parte de ustedes y aprender tanto.

Al todos mis compañeros de grupo, los de la 10 y los de la 2.

A todo mi grupo de la Lenin.

En fin gracias a todos los que han hecho posible esto.

Alejandro Murillo

A mis padres que han sido mi motor impulsor durante estos cinco años, sin los cuales no hubiera podido llegar hasta donde estoy.

A mi pequeño bebé Robertico, quien desde su nacimiento ha llenado mi corazón de amor y alegría.

A mi esposa Leidis por estar siempre conmigo en los buenos y malos momentos durante la carrera y por traer a la vida a mi más preciado tesoro.

A mi tutora Norbelis quien estuvo presente todo el tiempo guiándome y apoyándome para que todo saliera bien.

A mi hermana Lairobys de quien estoy orgulloso de ser hermano, la cual siempre se mantuvo aconsejándome y brindándome su amor y comprensión.

A mi sobrinita Ritchelie por quererme tanto y por portarse bien.

A mi cuñado Ricardo por tenernos siempre presente a mí y a mi familia.
A mi abuelita Estela y mi tío Humbertico por brindarme su cariño incondicionalmente.

A mis primos Yosbani, Yaidelin, Dalvita, Danaisy, Danae, Omarcito y Omairy quienes de una forma u otra me ayudaron y me tuvieron presente.

A mis tíos Walfrido, Daisy, Omar Lorenzo, Vivian, María del Carmen y Omar Ares por su disposición y entrega.

A mis suegros Ramón y Marlen por considerarme uno más de la familia.

A Mariela, Yoel, Margarita, Orly y Keilita por desear lo mejor para mí.

A mis amigos y profesores que durante la carrera me ayudaron y me guiaron.

A todos aquellos que directa o indirectamente aportaron su granito de arena en mi formación como profesional.

Alejandro Lorenzo

Resumen

En esta investigación se propone una estrategia de pruebas de software para ser utilizada en los proyectos productivos pertenecientes al Centro de Telemática de la Universidad de las Ciencias Informáticas. Se definen los conceptos relacionados con el proceso de Prueba de Software describiendo y clasificando sus niveles, tipos y técnicas.

Mediante entrevistas realizadas a especialistas en el desarrollo de software para las telecomunicaciones, se identifican las características de calidad significativas para los sistemas dedicados a esta rama. El principal objetivo de la estrategia es preservar dichas características mediante su aplicación. Igualmente se realiza un breve estudio sobre las metodologías de desarrollo usadas en el centro, con el fin de que la estrategia propuesta pueda ser usada en cada una de ellas.

La validación se realiza mediante el método Delphi, utilizando para esto una encuesta aplicada a 8 expertos en el campo de las pruebas de software, provenientes de distintas entidades.

Palabras Claves: Pruebas de Software, Estrategia de Pruebas, Software de Telecomunicaciones.

ÍNDICE

INTRODUCCIÓN

En las últimas décadas se ha experimentado un crecimiento vertiginoso a nivel mundial en la industria del software. En la actualidad, la mayoría de las empresas demandan informatizar procesos de vital importancia para su correcto funcionamiento, lo que ha desatado un gran auge de producción y exportación de software a lo largo del mundo, unido a una competencia cada vez más reñida en el mercado.

En los últimos años Cuba se ha propuesto incrementar la informatización de la sociedad, así como producir para la exportación distintos tipos de software. En este empeño se crea la Universidad de las Ciencias Informáticas (UCI) en el año 2002. Dentro de esta se han conformado Centros de Producción de Software para las distintas temáticas y otros servicios informáticos. (1)

Este es un mercado muy competitivo a nivel mundial por la diversidad de compañías dedicadas al sector (2), razón por la cual resulta complejo insertarse en el mismo, principalmente para un país pequeño con las limitaciones económicas que presenta. Tanto para garantizar la adecuada inserción en dicho mercado, como para la informatización de la sociedad con vistas a mejorar la eficiencia en producciones y servicios, la calidad de software es un factor imprescindible.

La garantía de la calidad del software es un concepto de vital importancia a la hora de iniciar un proyecto para que avance y cumpla los objetivos esperados. Es una "actividad de protección" que se aplica a lo largo de todo el proceso de Ingeniería del Software, la cual engloba: métodos y herramientas de análisis, diseño, codificación y prueba; revisión de técnicas formales que se aplican durante cada paso; estrategia de prueba multiescalada; control de la documentación del software y de los cambios realizados; procedimientos que aseguren un ajuste a los estándares de desarrollo del software; y mecanismos de medida y de información. (3)

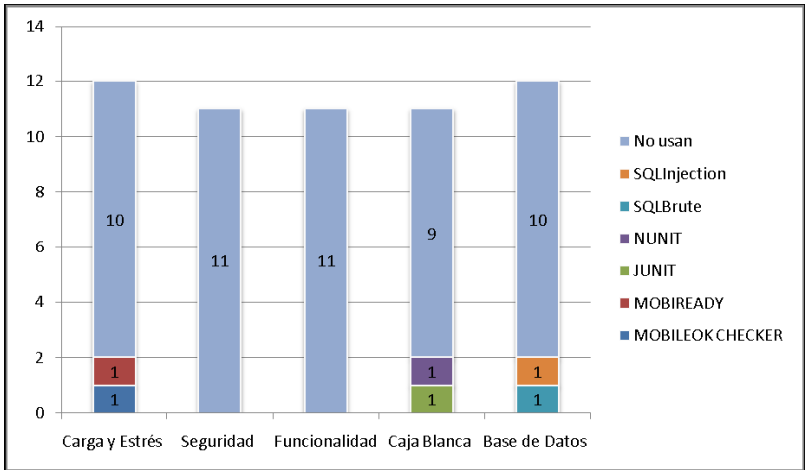
Entre las normas que establecen directrices para las actividades de calidad, se encuentra la NC ISO/IEC 9126-1:2005, la cual define diferentes características de calidad para los productos de software, brindando una consistente terminología sobre la calidad del producto y proporcionan un marco para especificar los requisitos de calidad.

Las aplicaciones de software han crecido en complejidad y tamaño, y por consiguiente también en costos (1). Hoy en día es crucial verificar y evaluar la calidad de lo construido con el objetivo de minimizar el costo de su reparación. Cuanto antes se detecte una falla, más barata es su corrección. Formando parte de las acciones que tributan a la calidad de software, el proceso de prueba es clave a la hora de detectar errores o fallas.

En el resumen dado a conocer por el Centro de Calidad para Soluciones Tecnológicas (Calisoft) de la UCI en noviembre del 2009, se evidenció que durante las Revisiones realizadas a los proyectos productivos de la universidad, se detectaron 77 no conformidades relacionadas con el proceso de pruebas de software. (3)

La Facultad 2 no queda exenta de dichos problemas; durante un diagnóstico realizado por Calisoft en esa misma fecha, se identificó un escaso uso de herramientas para automatizar el proceso de pruebas. (4)

Figura 1 Uso de las herramientas de pruebas en los proyectos.



De esta manera, se ilustra la poca calidad en el proceso de pruebas, esto se debe a la escasa experiencia existente en proyectos desarrollados para las telecomunicaciones, a la falta de personal calificado en aseguramiento y control de la calidad, además no se cuenta con un estudio detallado de las pruebas que pueden ser adaptadas a este tipo de software.

En la UCI el Laboratorio Industrial de Pruebas de Software (LIPS) perteneciente a Calisoft es el encargado de la realización de pruebas a todos los software de los distintos Centros Productivos con que cuenta la universidad. Para ellos se realizan una serie de pruebas que son genéricas para todos los sistemas, sin tener en cuenta pruebas específicas que tributen a las capacidades más relevantes para el software de telecomunicaciones.

Debido a lo anteriormente expuesto se tiene como **Problema a Resolver:**

¿Cómo garantizar que el proceso de pruebas de software en los proyectos productivos del Centro de Telemática tribute a las características de calidad más relevantes para los sistemas dedicados a las Telecomunicaciones?

Teniendo como **Objeto de Estudio** de la presente investigación el proceso de pruebas de software y como **Campo de Acción** el proceso de pruebas de software que se realiza en el Centro de Telemática de la Facultad 2.

En aras de dar solución al problema mencionado se plantea el siguiente **objetivo general**: Definir una estrategia de pruebas de software adecuada a las características de los productos para las telecomunicaciones.

Teniendo como **Objetivos Específicos**:

1. Analizar de las características de calidad que propone la NC ISO/IEC 9126-1:2001 cuáles son las relevantes para el software destinado a las telecomunicaciones.
2. Analizar el proceso de pruebas de software en una representación de las empresas relacionadas con las telecomunicaciones.

3. Analizar las metodologías de desarrollo usadas en los proyectos del Centro de Telemática.
4. Identificar los niveles, tipos, métodos y técnicas de pruebas que tributen a las características seleccionadas.
5. Validar la estrategia propuesta mediante la valoración de expertos.

Planteándose como **idea a defender** que: al definir una estrategia de pruebas de software adecuada a las características de los sistemas para las telecomunicaciones, el proceso de pruebas de software en el Centro de Telemática contribuirá a garantizar a dichas características.

Como **tareas de investigación** se tienen:

- ✓ Estudiar la NC ISO/IEC 9126:2005.
- ✓ Entrevistar a especialistas en el desarrollo de software para las telecomunicaciones.
- ✓ Estudiar la bibliografía existente sobre pruebas de software.
- ✓ Visitar empresas relacionadas con el desarrollo de software para las telecomunicaciones.
- ✓ Conocer el funcionamiento de los proyectos productivos del Centro de Telemática.
- ✓ Indagar sobre las metodologías de desarrollo de software usadas en el Centro de Telemática.
- ✓ Definir los tipos de pruebas que tributan a las características de calidad resaltadas por los especialistas entrevistados.
- ✓ Definir los métodos de prueba y las técnicas a ser usados para llevar a cabo cada uno de los tipos de prueba definidos.

- ✓ Realizar encuesta a expertos previamente seleccionados con experiencia en el área de pruebas de software.

Métodos de Investigación:

Entre los Métodos Científicos teóricos existentes se utilizó el Analítico - Sintético, que se caracteriza por buscar la esencia de los fenómenos, los rasgos que lo caracterizan y distinguen, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio. En este caso se han extraído las características más importantes del software dedicado a las telecomunicaciones para seleccionar las pruebas adecuadas a estas características.

Dentro de los Métodos Empíricos se empleó la entrevista, que no es más que una conversación planificada, en este caso se empleó para conocer de las características que explica las NC ISO/IEC 9126-1:2005 cuáles son las más importantes para los software de telecomunicaciones, para ello se entrevistaron especialistas en el desarrollo de software para esta rama. También se entrevistaron a los líderes de los proyectos del Centro para conocer la dinámica del desarrollo y las metodologías utilizadas.

El trabajo fue dividido en tres capítulos. A continuación se muestra el nombre de cada uno con una breve explicación de su contenido:

Capítulo 1: Fundamentación teórica. Contiene los conceptos significativos, la relación entre las pruebas y la calidad de un software, los tipos de pruebas existentes y las usadas a nivel mundial en el área de las telecomunicaciones. También su uso en Cuba y específicamente en la UCI.

Capítulo 2: Propuesta de la Estrategia de Pruebas. Está dedicado principalmente a describir la estrategia de pruebas propuesta partiendo de la relación entre las características de calidad de más relevancia para el software dedicado a las telecomunicaciones y los tipos de pruebas encaminadas a estas características.

Capítulo 3: Validación de la Propuesta. Se valida la estrategia propuesta utilizando el método Delphi.

Capítulo 1: Fundamentación Teórica.

Introducción

Los niveles, tipos y técnicas de prueba nos ayudan a comprender las distintas formas de realizar las pruebas, este capítulo tiene como propósito tratar la relación de dichos conceptos. Se realiza un análisis de la NC ISO/IEC 9126-1:2005, la cual describe las características de calidad que todo software debe tener. En el mismo también se explica en qué consisten las Evaluaciones de Software y se realiza un estudio de las Metodologías de Desarrollo de Software utilizadas en el Centro de Telemática. Además, se da a conocer cómo se encuentra la situación de las Pruebas de Software en el Área de las Telecomunicaciones en el contexto mundial y en Cuba.

1.1 Introducción a la Calidad del Software.

1.1.1 Calidad del software.

Partiendo de que la calidad es la concordancia con los requisitos funcionales y de rendimiento, explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente. (5) Las pruebas de software nos permiten detectar la mayor cantidad de fallas posibles en los productos antes de ser entrados al usuario.

1.1.2 Pruebas de Software.

Una prueba de software es un proceso en el cual se ejecuta un sistema o uno de sus componentes en circunstancias previamente especificadas, los cuales se registran, se analizan y se realiza una evaluación a partir de los resultados obtenidos, con el objetivo final de limar asperezas en el sistema de producción y prevenir así la acumulación de trabajo deficiente.

La prueba del software es un elemento crítico para la garantía de la calidad del software. Su objetivo "es diseñar pruebas que saquen a la luz diferentes clases de

errores con la menor cantidad de tiempo y espacio" (5). De esta manera, se contribuiría a preservar las características de calidad que distinguen al software en desarrollo.

1.2 Norma NC ISO/IEC 9126-1:2005

La norma NC ISO/IEC 9126:2005 es un estándar internacional para la evaluación del software. Está dividida en 4 partes que tratan respectivamente los siguientes temas: modelo de calidad, métricas externas, métricas internas y métricas para la calidad en uso. Esta investigación está centrada en el estudio del primero de estos acápites, debido a que se analizarán, entre las características propuestas en el modelo de calidad, las significativas para los sistemas dedicados a las telecomunicaciones, no siendo objetivo de esta investigación profundizar en los otros acápites ya que están relacionados con las métricas del software. El modelo de calidad establecido en la primera parte del estándar, NC ISO/IEC 9126-1:2005, clasifica la calidad del software en un conjunto estructurado de características y sub-características de la siguiente manera (6):

Funcionalidad: se define como un conjunto de atributos que atañen a la existencia de un conjunto de funciones y sus propiedades específicas. Estas funciones son las que satisfacen las necesidades implícitas y establecidas. Esta característica del software puede ser desglosada en varias sub-características:

- ✓ Adecuación: no es más que la capacidad del software de proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos del usuario.
- ✓ Exactitud: puede definirse como la capacidad del software para proporcionar resultados correctos o que necesitan un determinado grado de precisión.
- ✓ Interoperabilidad: es la capacidad del software de interaccionar con uno o más sistemas especificados.

- ✓ Seguridad: definida como la capacidad del software de proteger la información y los datos.

Confiabilidad: se define como el conjunto de atributos que atañen a la capacidad del software para mantener su nivel de prestación bajo condiciones establecidas durante un tiempo establecido. Se descompone en las siguientes características:

- ✓ Madurez: es la capacidad del software para evitar fallos como resultados de defectos del software.
- ✓ Tolerancia a fallos: capacidad del software para mantener un nivel especificado de rendimiento en casos de fallos del software.
- ✓ Capacidad de recuperación: capacidad para restablecer el nivel de rendimiento y de recuperación de datos afectados directamente en el caso de un fallo.

Facilidad de uso: se entiende como la capacidad del producto software de ser entendido, aprendido, usado y atraer al usuario, cuando es utilizado bajo ciertas condiciones específicas. Se descompone en:

- ✓ Fácil comprensión: se entiende como la capacidad del software que permite al usuario si el producto es aceptable y cómo puede ser usado para tareas particulares y determinadas condiciones de uso.
- ✓ Fácil aprendizaje: capacidad del producto software que permite al usuario aprender la aplicación software.
- ✓ Operatividad: capacidad del producto software que permite al usuario controlar y usar la aplicación software.
- ✓ Software atractivo: capacidad del producto software de ser atractivo al usuario.

Eficiencia: está dada por la capacidad del producto software para proporcionar un rendimiento apropiado relacionado con el total de recursos utilizados bajo condiciones establecidas. Se subdivide en las siguientes características.

- ✓ Comportamiento frente al tiempo: capacidad del producto software para proporcionar una respuesta y un tiempo de procesamiento apropiados al desarrollar sus funciones bajo condiciones establecidas.
- ✓ Uso de recursos: capacidad del producto software para utilizar un apropiado número de recursos y tiempo de ejecución cuando el software desarrolla sus funciones bajo condiciones establecidas.
- ✓ Adherencia a normas: capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes en leyes o prescripciones similares.

Mantenibilidad: se define como la capacidad del producto software para ser modificado. Se descompone en las siguientes características.

- ✓ Facilidad de análisis: capacidad del producto software para diagnosticar deficiencias o causas de fallos en el software.
- ✓ Capacidad para cambios: capacidad del producto software que permite la ejecución de una modificación específica en ella misma.
- ✓ Estabilidad: capacidad del producto de software para evitar defectos no esperados debidos a modificaciones en el mismo.
- ✓ Facilidades para pruebas, capacidad del producto software que permite al software que ha sido modificado, ser evaluado.

Portabilidad: se entiende como la capacidad del producto software para ser transferido de un entorno a otro. El entorno se interpreta tanto a nivel software y hardware, como aquel entorno relacionado con la organización. Se divide en:

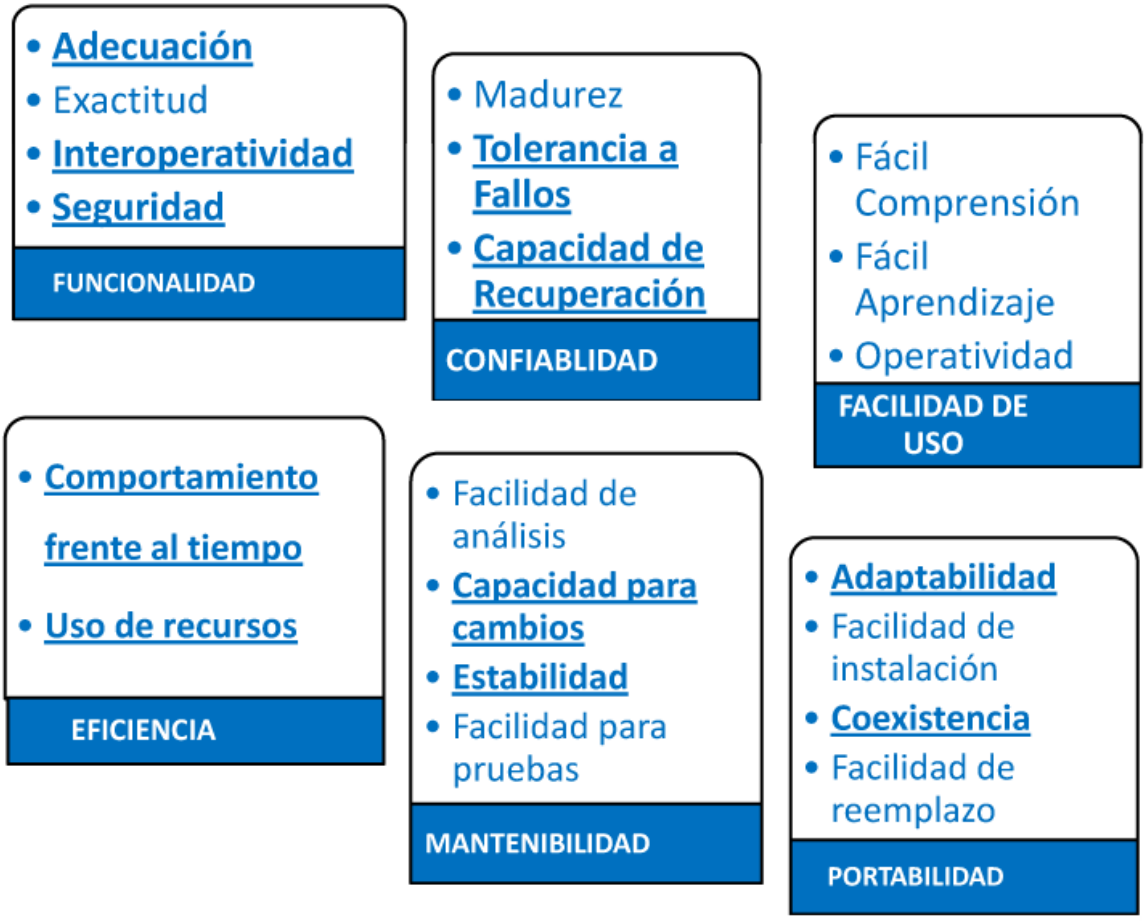
- ✓ Adaptabilidad: capacidad del producto software para ser adaptado a diferentes entornos especificados sin aplicar acciones alejadas de aquellas que el propio software proporcione.
- ✓ Facilidad de instalación: capacidad del producto software para ser instalado en un entorno específico.
- ✓ Coexistencia: capacidad del producto software de coexistir con otros programas independientes en un entorno común y compartiendo recursos también comunes.
- ✓ Facilidad de reemplazo, capacidad del producto software de ser utilizado en lugar de otro producto software específico para el mismo propósito que éste y en un entorno similar.

Para identificar cuáles de estas características son más significativas específicamente para el software de telecomunicaciones se entrevistaron especialistas que han estado ligados al desarrollo de este tipo de software.

1.2.1 Análisis de las Entrevistas realizadas a Especialistas en el Sector de las Telecomunicaciones.

Las entrevistas fueron realizadas a especialistas con experiencia acumulada durante su trabajo en el desarrollo de software para las telecomunicaciones, con el fin de adquirir conocimientos cualitativos sobre este tipo de sistema, mediante una serie de preguntas previamente planificadas (Anexo 1) se pudo conocer su opinión acerca de cuáles son las características de la NC ISO/IEC 9126-1:2005 significativas para el software desarrollado para las telecomunicaciones y de manera específica cuales de las sub-características presentes en la misma presentan relevancia para estos sistemas, un resumen de estas características señaladas puede verse en la Figura 1.1.

Fig. 1.1 Características y sub-características de la NC ISO/IEC 9126-1:2005



Entre las características enumeradas como más importantes por los especialistas estuvo en primer orden la funcionalidad y entre sus sub características la seguridad, debido al manejo de información sensible que tienen muchos software dedicados a las telecomunicaciones como pueden ser información de cuentas bancarias, debido al desarrollo de la telefonía móvil y otros dispositivos portátiles muchas operaciones de compra y venta hoy en día se hacen desde los mismos. También en el ámbito de la seguridad es importante proteger la información generada en el manejo de centrales telefónicas debido a la cantidad de usuarios que pudiera afectar cualquier ataque contra estos sistemas, además de dejar vulnerable la seguridad del país, debido al posible colapso de sus sistemas de comunicación.

Dentro del mismo acápite de la funcionalidad los especialistas señalaron la interoperabilidad como otra de las características de vital importancia para este tipo de software. Debido a que muchos de estos sistemas tienen que interactuar con otros sistemas para intercambiar información.

La confiabilidad fue otra de las características señaladas en las entrevistas realizadas. Para las telecomunicaciones esta reviste una vital importancia con el fin de mantener operativa y en servicio cualquier red de comunicaciones. Dentro de la confiabilidad resaltan características como son la tolerancia a fallos, este tipo de software tiene que lograr mantener un nivel de rendimiento en casos que se produzcan fallos. Igualmente la capacidad de recuperación es importante para que ante cualquier fallo del sistema, este pueda recuperarse en el menor tiempo posible a fin de restablecer el servicio que este presta.

Otra de las características presentes en la NC ISO/IEC 9126-1:2005 resaltada por los especialistas entrevistados fue la eficiencia, basándose en la necesidad de este tipo de sistemas de trabajar aprovechando al máximo los recursos de hardware disponibles para dar servicio a la mayor cantidad de usuarios posibles. Como subcaracterísticas de la eficiencia se pueden destacar el comportamiento frente al tiempo, que es un eslabón crítico para un gran número de software de telecomunicaciones debido a la necesidad de una rápida respuesta al cliente. Además, el uso de recursos es considerada una sub característica relevante dentro de la eficiencia, debido al alto costo de los equipos de hardware utilizados para este tipo de sistema, estos deben funcionar de la manera esperada con la menor cantidad de recursos posibles y haciendo un uso eficiente de los mismos.

La mantenibilidad también resultó una de las características mencionadas por los especialistas consultados. La rapidez con que un sistema de telecomunicaciones pueda ser actualizado o realizadas otras reparaciones que sean necesarias son de gran importancia para tener fuera de funcionamiento al sistema el menor tiempo posible. Es por eso que esta característica fue resaltada, debido a que un medidor de la calidad para los sistemas de telecomunicaciones es que se encuentre el menor tiempo posible fuera de servicio.

La última de las características señalada por los especialistas fue la portabilidad. En el mundo existe una gran tendencia a un mayor uso de dispositivos portátiles. Es por eso que el tipo de software analizado debe poseer una alta capacidad de portabilidad para adaptarse a dichos dispositivos.

Para garantizar la calidad del software del centro la evaluación del mismo debe ser integral, y se debe contar con una estrategia de prueba capaz de garantizar la comprobación de cada una de estas características enumeradas por los especialistas.

1.3 Metodologías de Desarrollo de Software.

Desarrollar un software con calidad depende de un sin número de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo en un determinado proyecto, es trascendental para el éxito del producto. Las metodologías de desarrollo de software abarcan todo el ciclo de vida del software, y se definen como "un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software". (7)

En los últimos tiempos la cantidad y variedad de las metodologías de desarrollo de software han aumentado de forma impresionante. Se podría decir que en estos últimos años se han desarrollado 2 corrientes en lo referente a las metodologías, las llamadas metodologías pesadas y las metodologías ligeras. La diferencia fundamental entre ambos radica en que mientras las metodologías pesadas intentan conseguir el objetivo común por medio del orden y documentación, las ligeras o ágiles tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso (8). En este acápite se hará referencia al proceso de pruebas en 3 de estas metodologías, Proceso Unificado de Desarrollo de Software (del inglés Rational Unified Process, RUP), sXP metodología creada en la UCI que utiliza procesos de SCRUM y XP, además del Proceso Guiado por Funcionalidad (del inglés Feature

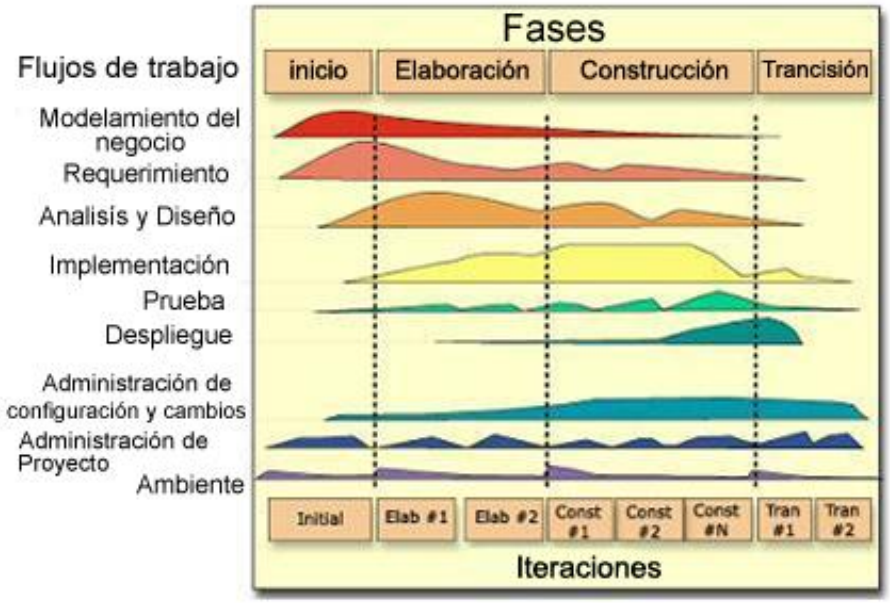
Driver Development, FDD) debido a que estas son las usadas por los distintos proyectos productivos del Centro de Telemática.

1.3.1 Proceso Unificado de Desarrollo de Software.

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. (9)

RUP está dividido en fases y flujos de trabajo que unidos conforman el ciclo de vida del proyecto. Las fases son inicio, elaboración, construcción y transición, y los flujos de trabajo que propone son modelamiento del negocio, requerimientos, análisis y diseño, implementación, pruebas, despliegue, administración de configuración y cambios, administración del proyecto y ambiente, como puede verse en la Figura. 1.2. (10)

Figura 1.2 Proceso Unificado de Desarrollo de Software.



Esta investigación está enfocada principalmente en el flujo de trabajo de pruebas, que como se muestra en la Figura 1.2 debe de estar presente en cada una de las fases ciclo de desarrollo ya sea en mayor o menor grado. RUP plantea para el flujo de trabajo de pruebas una serie de objetivos (10):

- ✓ Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas del sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- ✓ Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando los procedimientos de prueba que especifican cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- ✓ Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

Para cumplir estos objetivos RUP propone una serie de trabajadores y artefactos, entre los trabajadores que toman parte en este flujo de trabajo se encuentran: diseñador de pruebas, ingeniero de componentes, ingeniero de pruebas de integración e ingeniero de prueba de sistemas. Estos trabajadores interactúan con una serie de artefactos.

1.3.1.1 Artefactos de RUP para el Flujo de Trabajo de Pruebas.

Modelo de Pruebas.

El artefacto Modelo de Prueba describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. Puede describir también cómo se han probado aspectos

específicos del sistema, por ejemplo, si la interfaz de usuario del sistema cumple con su objetivo y describe el cumplimiento de los requisitos funcionales y no funcionales del sistema. Es una colección de casos de pruebas, procedimientos de prueba y componentes de prueba.

Caso de Prueba.

El diseño de casos de prueba es uno de los pasos más importantes al realizar una estrategia de pruebas, ya que estos constituyen la fuente para evaluar los resultados del software. Especifican una forma de probar el sistema, incluyendo la entrada o resultado con que se va a probar y las condiciones bajo las que ha de probarse. Una de las principales características que deben presentar los casos de prueba es su carácter abarcador, es decir, deben ser completos y estar adaptados al producto, pues deben ofrecernos la posibilidad de encontrar la mayor cantidad de errores posibles en un tiempo y costo no muy elevados.

Procedimiento de Prueba.

Un Procedimiento de Prueba especifica cómo realizar uno o varios casos de pruebas o partes de estos. Puede ser una instrucción para un individuo sobre cómo realizar un caso de prueba manualmente, o una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas, para crear componentes ejecutables de prueba.

Componente de Prueba.

Un Componente de Prueba automatiza uno o varios procedimientos de prueba o partes de ellos. Estos pueden ser desarrollados utilizando un lenguaje de guiones o un lenguaje de programación, o pueden ser grabados con una herramienta de automatización de pruebas. Los componentes de pruebas se utilizan también para probar los componentes en el modelo de implementación, proporcionando entradas de pruebas, controlando y motorizando la ejecución de los componentes a probar e informando de los resultados de las pruebas. Pueden ser implementados usando tecnología de objetos.

Plan de prueba.

La construcción de un buen Plan de Pruebas es el principal factor de éxito para la puesta en práctica de una estrategia de pruebas que permita entregar un software de mejor nivel. Es el artefacto que permite trazar el tipo de prueba que se le va a aplicar al producto, cuyo propósito es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario y los responsables del proceso de pruebas. Durante el desarrollo del software se diseña el Plan de Prueba con el objetivo de asegurar que todos los requisitos tanto funcionales como de rendimiento, se satisfagan.

Defecto.

Un defecto es un síntoma de un fallo en el software o un de un problema descubierto en una revisión, el cual puede ser utilizado para localizar cualquier cosa que los desarrolladores necesiten registrar como síntoma de problema en el sistema.

Evaluación de Prueba.

Es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, de código y el estado de los defectos. La evaluación es realizada por los diseñadores quienes comparan los resultados obtenidos con los objetivos trazados en el Plan de Prueba. Durante la evaluación de las pruebas, se realizan métricas que permiten determinar el nivel de calidad del software y qué cantidad de pruebas se deben realizar. (10)

Luego de expuestas las principales características de la metodología RUP, se puede concluir que la misma tiene como objetivos, asegurar la producción de un software de calidad dentro de plazos y presupuestos predecibles, por lo que requiere de cierta preparación así como de disponer de recursos para aplicarla, en el tiempo y el costo predefinidos dentro del proyecto. También se puede asegurar que es una metodología aplicable a proyectos que posean gran cantidad de personal, tiempo para su desarrollo y un presupuesto que permita cubrir el

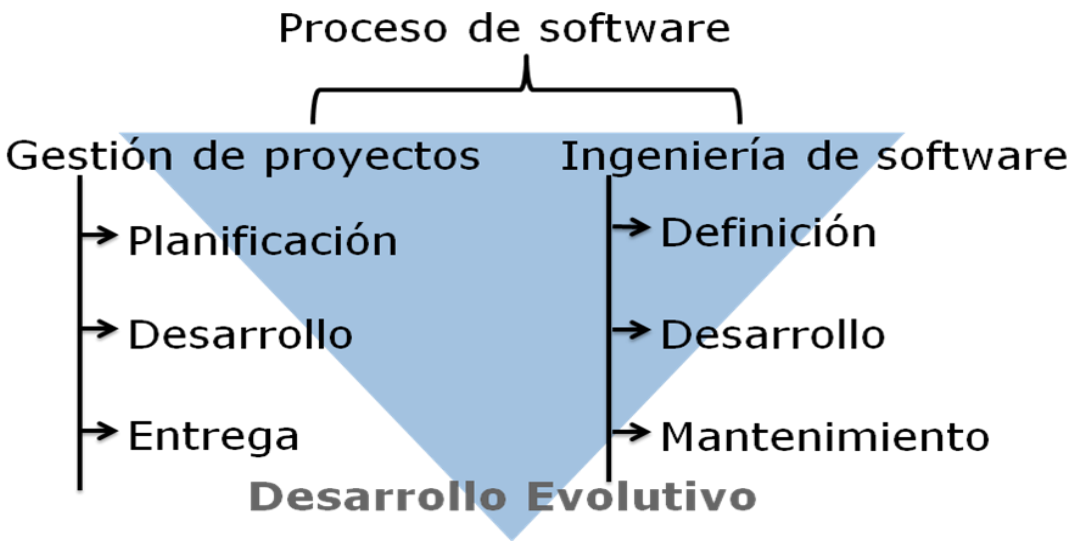
desarrollo del mismo. Para los proyectos pequeños y en los cuales se necesite una gran adaptabilidad, esta metodología no es recomendable, en estos casos se recomienda usar metodologías ágiles, de estas las usada en los proyectos del Centro de Telemática son sXP y FDD.

1.3.2 sXP.

sXP es una metodología de desarrollo de software creada en la facultad 10 de la UCI. En esta se usa SCRUM para la planificación de los proyectos que usarán métodos ágiles como metodología para su proceso de desarrollo, pues la misma es una forma de gestionar proyectos de software, no es una metodología de análisis, ni de diseño, es una metodología de gestión del trabajo. El rol propuesto por esta metodología para el proceso de pruebas es el encargado de pruebas. (11)

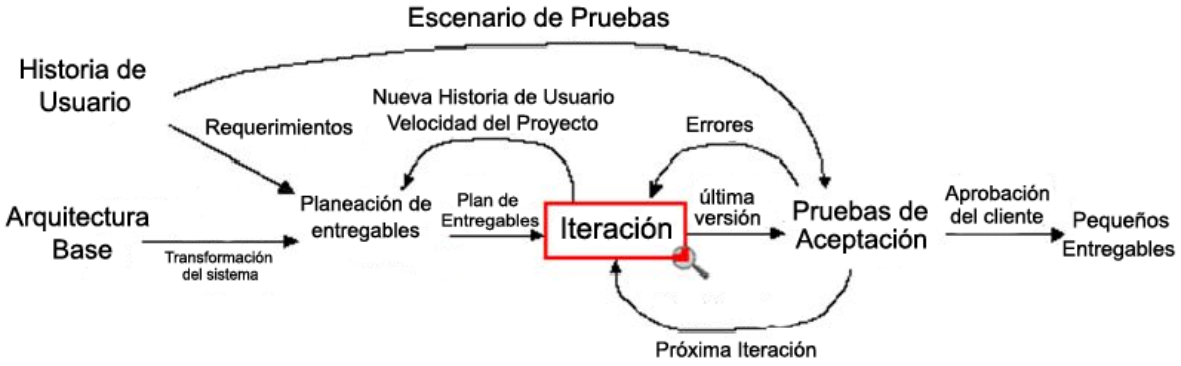
La labor de encargado de pruebas es ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Figura 1.3: Metodología sXP.



La Programación Extrema (del inglés Extreme Programming, XP) es una de las metodologías desarrollo de software ágil más usadas para proyectos de corto plazo y equipos pequeños. Se le llama extrema, porque define pocas reglas y pocas prácticas a través de la eliminación de procesos y técnicas que carecen de valor. En el caso de los proyectos que trabajan con XP, deben seguir procesos disciplinados, pero más que eso, deben combinar la disciplina con la adaptabilidad necesaria del proceso. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. XP, dadas sus características, es la metodología más apropiada para un entorno caracterizado por requerimientos cambiantes originados mayormente por un mercado inestable y el inminente avance de la tecnología y los negocios, por tanto, es recomendable aplicarla en caso de que los requisitos estén vagamente definidos y los clientes se pueden involucrar la mayor parte del tiempo en el desarrollo del proyecto.

Figura 1.4: Metodología XP.

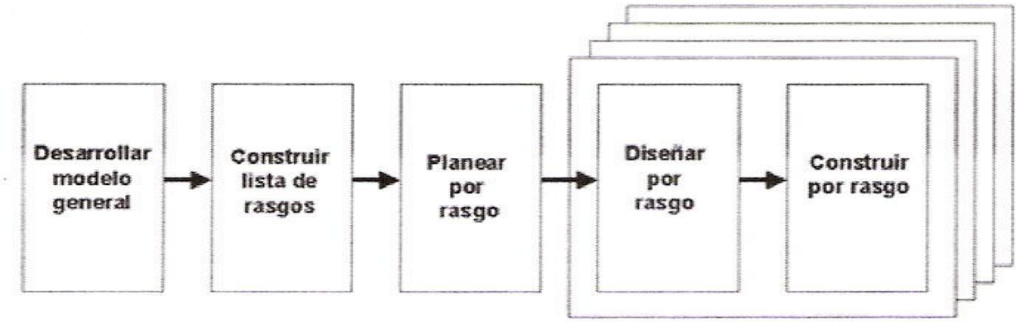


La producción de código en proyectos que usen esta metodología está dirigida por las Pruebas Unitarias. Las Pruebas Unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las Pruebas Funcionales para cada Historia de Usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial. (11)

1.3.3 Proceso Guiado por Funcionalidad

FDD es un proceso diseñado por Peter Coad, Erich Lefebvre y Jeff De Luca y se podría considerar a medio camino entre RUP y XP, aunque al seguir siendo un proceso ligero es más similar a este último. Está pensado para proyectos con tiempo de desarrollo relativamente corto. Se basa en un proceso iterativo con iteraciones cortas que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar. (12)

Figura 1.5: Metodología FDD.



Las iteraciones se deciden en base a funcionalidades (*features*), que son pequeñas partes del software con significado para el cliente. Un proyecto en FDD se divide en 5 fases:

1. Desarrollo de un modelo general.
2. Construcción de la lista de funcionalidades.
3. Plan de releases en base a las funcionalidades a implementar.
4. Diseñar en base a las funcionalidades.
5. Implementar en base a las funcionalidades.

Las funcionalidades a implementar en un *release* se dividen entre los distintos subgrupos de equipo, y se procede a implementarlas. Las clases escritas tienen propietarios, solo puede cambiarlas quien las crea, por ello en el equipo que implementa una funcionalidad dada deberán estar todos los dueños de las clases implicadas, pudiendo encontrarse un programador en varios grupos, implementando distintas funcionalidades. Habrá también un programador jefe que hará las funciones de líder del grupo que implementa esa funcionalidad.

FDD propone 3 divisiones o jerarquías para sus roles, roles claves entre los que se encuentran director del proyecto, arquitecto jefe, director de desarrollo, programador jefe, propietarios de clases y experto del dominio. Como parte de los roles de soporte se encuentran administrador del dominio y administrador de

realease. Dentro de los roles adicionales tenemos el probador y el escritor de documentos técnicos.

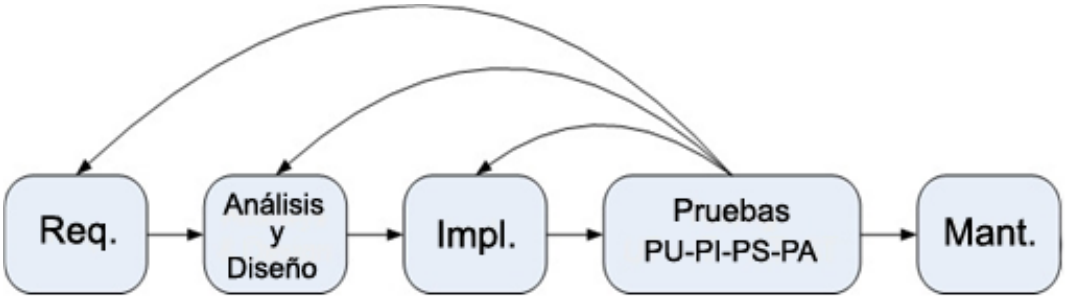
El probador es el encargado de diseñar y ejecutar las pruebas a realizar al sistema y de conjunto con el escritor de documentos técnicos llevar los artefactos generados durante las pruebas. (12)

1.3.4 Pruebas en Metodologías Convencionales y Ágiles: Papeles diferentes.

1.3.4.1 Las Pruebas en los enfoques convencionales.

Cuando se habla de las pruebas en los enfoques convencionales, fundamentalmente se identifican cuatro elementos relacionados: Pruebas Unitarias, Pruebas de Integración, Pruebas de Aceptación y Pruebas de Sistema. En estas metodologías se presta especial atención a la elaboración de la especificación de requisitos software. Dicha especificación se refina dentro del proceso de identificación de los requisitos del software mediante iteraciones. En este enfoque, el proceso de desarrollo se dirige desde la especificación de requisitos hacia el código y posteriormente desde el código hasta las pruebas. En consecuencia, aunque en las metodologías convencionales las pruebas son importantes, estas dependen directamente del resto de actividades del proceso de desarrollo. Y por lo tanto, en estos casos, el proceso de validación se entiende como un complemento del proceso de desarrollo. (13)

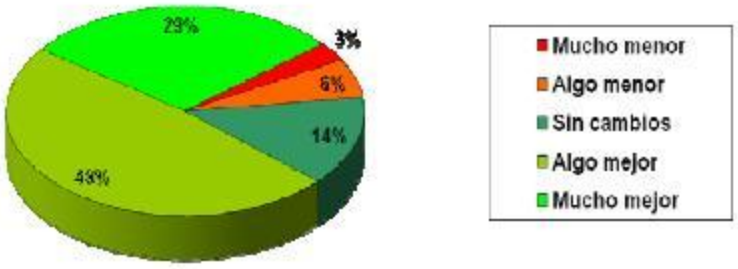
Figura 1.6: Pruebas en Metodologías Convencionales.



1.3.4.2 Las Pruebas en Metodologías Ágiles.

Las metodologías ágiles en general, y particularmente XP, son de alguna manera los responsables del aumento de popularidad de las pruebas del software. En XP las necesidades de los usuarios se representan mediante "Historias de Usuario" que describen los requisitos del sistema. Cada Historia de Usuario lleva asociada criterios de aceptación y para cada uno de ellos se definen casos de prueba. Las Pruebas de Aceptación se escriben en las fases tempranas del desarrollo, y antes de que el sistema se implemente, para validar las necesidades de los usuarios y para dirigir la implementación. Se puede considerar que las Historias de Usuario y, por extensión, las pruebas asociadas juegan el papel de especificaciones del sistema. En los enfoques ágiles las pruebas son el centro de la metodología y, por lo tanto, son ellas las que dirigen el proceso de desarrollo. Las metodologías ágiles plantean que el desarrollo no es un conjunto de fases en las que las pruebas son una fase más, sino que abogan porque las prácticas y el desarrollo estén completamente integradas, lo que puede llevar a modificar las estructuras organizativas de las empresas (14). Según datos consultados (15-16) se puede comprobar que cerca del 70% de las empresas ya están incorporando algunas prácticas ágiles en su proceso de desarrollo. Según estos autores, esto ha traído como consecuencia una mejora de la calidad de los productos entregados en casi un 70% de los proyectos, como puede verse en la Figura 1.7.

Figura 1.7: Impacto de las metodologías ágiles en la calidad. (15)



1.4 Lenguajes de Programación

En el Centro de Telemática el lenguaje que predomina es Java, este es usado en el mundo de las telecomunicaciones debido a su adaptabilidad a distintos dispositivos y a ser multiplataforma. Otra de las ventajas que presenta es ser sumamente robusto ya que fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria. Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la red, la seguridad se impuso como una necesidad de vital importancia. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real. (17)

Otros lenguajes también son usados en el centro ellos son PHP y Phyton, el primero es un lenguaje de programación web, es el acrónimo de *Hipertext Preprocesor*. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. (18)

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo,

lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

- ✓ La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- ✓ La sencillez y velocidad con la que se crean los programas.
- ✓ La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.

Además, Python es gratuito, incluso para propósitos empresariales. (19)

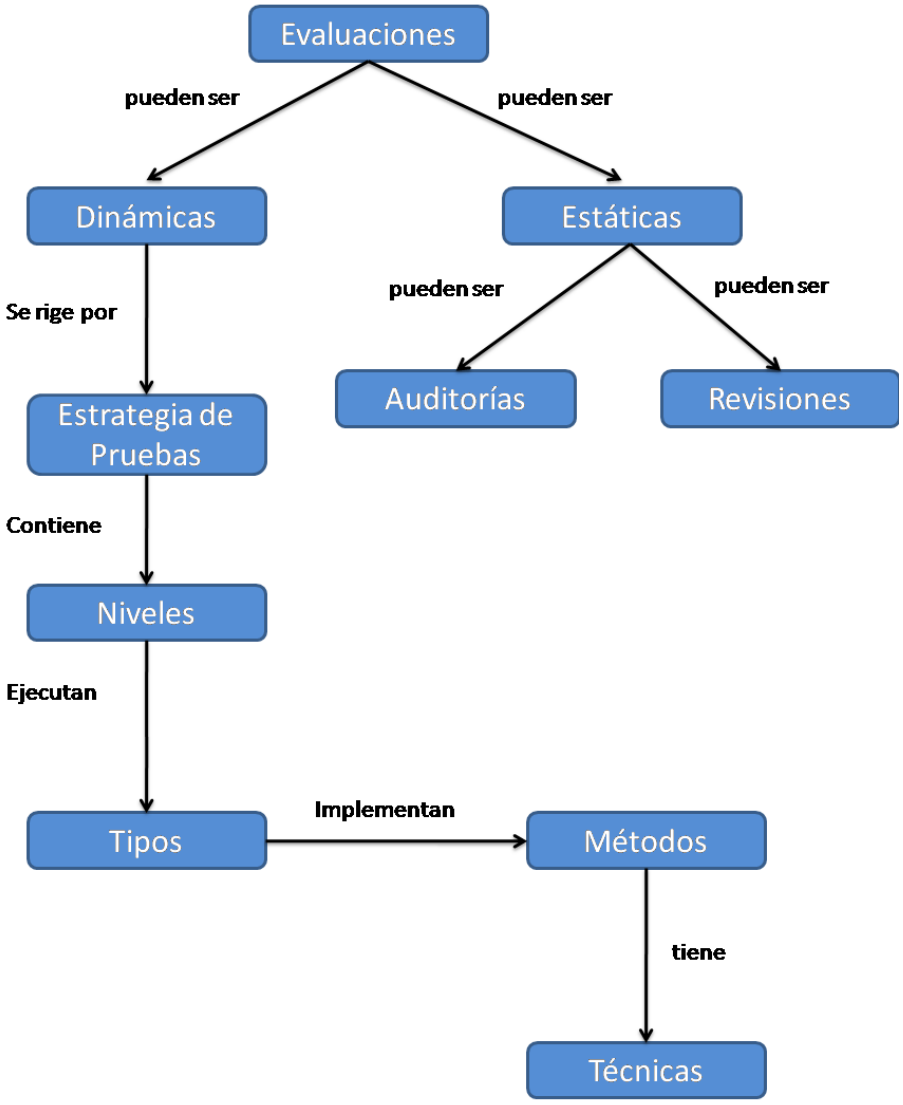
1.5 Evaluaciones de Software.

Durante todo el ciclo de vida de un proyecto de software se realizan evaluaciones para comprobar los diferentes artefactos generados durante su desarrollo. Las evaluaciones pueden ser estáticas o dinámicas. Las evaluaciones estáticas son las realizadas a todos los artefactos que se van generando durante las distintas fases del desarrollo del sistema, estas se pueden dividir en auditorías y revisiones. Las evaluaciones dinámicas, también llamadas pruebas de software, son las encargadas de comprobar la idoneidad de un sistema para cumplir con la función para la cual ha sido desarrollado. El fin principal de esta etapa es identificar los defectos que pudieran comprometer el buen funcionamiento del software y asegurar que se han corregido antes de entregar el producto final al cliente. La etapa de prueba software debe regirse por una estrategia de pruebas, la cual describe el enfoque y los objetivos generales de esta etapa. (20)

Dentro de la bibliografía consultada se pueden observar dos criterios diferentes con respecto a la clasificación de las técnicas o métodos de prueba. Según Pressman (5) se definen como métodos las pruebas de Caja Blanca, Caja Negra y

las Basadas en Experiencia, mientras que en la estrategia de evaluación de software de Calisoft (20) estas se definen como técnicas de prueba en lugar de métodos. A lo largo de la investigación se hará mención a las pruebas de Caja Blanca, Caja Negra y Basadas en Experiencia como Métodos de Prueba como lo define Pressman (5), igualmente de esta forma lo define Garbajosa (13). También se definen como métodos de pruebas en la conferencia que sobre el tema se imparte en la universidad. (21)

Figura 1.8: Evaluaciones al Software.



1.5.1 Estrategia de Pruebas.

Como se muestra en la Figura 1.8, la estrategia de prueba define además de los objetivos generales de las pruebas y su enfoque, los niveles de prueba a ser ejecutados en concordancia con las características del proyecto en cuestión. En cada uno de los niveles la estrategia debe definir qué tipos de pruebas se van a realizar así como métodos y las técnicas a implementar para cada uno de ellos.

1.5.2 Niveles de Pruebas.

Los niveles se clasifican en cuanto a 2 criterios, el primero de ellos define a qué elemento se le realiza la prueba, estos son Nivel de Unidad o Prueba Unitaria, Nivel o Prueba de Integración y Nivel o Prueba de Sistema. El otro criterio define quién es el encargado de hacer las pruebas, dentro de estos se enumeran Pruebas de Desarrollador, Pruebas Independientes y Pruebas de Aceptación (20).

1.5.2.1 Nivel de Unidad.

Es la prueba enfocada a los elementos probables más pequeños del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a pruebas de Caja Blanca. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido.

Prueba de unidad en el contexto Orientado a Objeto

En lugar de módulos individuales, una menor unidad a probar es la clase u objeto encapsulado. Una clase puede contener un cierto número de operaciones, y una operación particular puede existir como parte de un número de clases diferentes. Esta prueba de clases para el software Orientado a Objeto es equivalente a la prueba de unidad para el software convencional.

La prueba de clases para software Orientado a Objeto está dirigida por las operaciones encapsuladas por la clase y el estado del comportamiento de la clase. No se puede probar una operación aisladamente sino como parte de una clase [8].

1.5.2.2 Nivel de Integración.

Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un requisito. Esta prueba debe ser responsabilidad de desarrolladores y de independientes, sin solaparse. Es el proceso de combinar y probar múltiples componentes juntos. El objetivo es tomar los componentes probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño. Se llama integración incremental cuando el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir.

Prueba de Integración en el contexto Orientada a Objeto

Debido a que el software Orientado a Objeto no tiene una estructura de control jerárquica, las estrategias convencionales de integración ascendente y descendente poseen un significado poco relevante en este contexto.

Generalmente se pueden encontrar dos estrategias diferentes de pruebas de integración en sistemas Orientado a Objeto. La primera, prueba basada en hilos (*threads*), integra el conjunto de clases necesarias para responder a una entrada o evento del sistema. Cada hilo se integra y prueba individualmente. El segundo enfoque para la integración, es la prueba basada en el uso. Esta prueba comienza la construcción del sistema integrando y probando aquellas clases (llamadas clases independientes) que usan muy pocas de las clases. Después de probar las clases independientes, se comprueba la próxima capa de clases, llamadas clases dependientes, que usan las clases independientes. Esta secuencia de capas de pruebas de clases dependientes continúa hasta construir el sistema por completo (21).

1.5.2.3 Nivel de Sistema.

Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

Para asistir en la determinación de casos de prueba de sistema, el ejecutor de la prueba debe basarse en los casos de uso del sistema. El caso de uso brinda un escenario que posee una alta probabilidad con errores encubiertos en los requisitos de interacción del cliente. Los métodos convencionales de prueba de caja negra, pueden usarse para dirigir estas pruebas. (21)

1.5.2.4 Prueba de Desarrollador.

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para el Nivel de Unidad, aunque en la actualidad en algunos casos pueden ejecutarse en el Nivel de Integración. (21)

1.5.2.5 Prueba Independiente.

Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores. El objetivo de esta prueba es proporcionar una perspectiva diferente y en un ambiente más rico que los desarrolladores. Una vista de la prueba independiente es la prueba independiente de los stakeholder, que son pruebas basadas en las necesidades y preocupaciones de los stakeholders. (21)

1.5.2.6 Prueba de Aceptación.

La prueba de aceptación del usuario es la prueba final antes del despliegue del sistema. Durante su desarrollo se valida que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponde al usuario.

La validación de las funcionalidades del sistema se consigue mediante la realización de pruebas de Caja Negra que demuestran la conformidad con los requisitos y que se recogen en el Plan de Pruebas, el cual define las verificaciones a realizar y los casos de prueba asociados. Dicho plan está diseñado para asegurar que se satisfacen todos los requisitos funcionales especificados por el usuario teniendo en cuenta también los requisitos no funcionales relacionados con el rendimiento, seguridad de acceso al sistema, a los datos y procesos, así como a los distintos recursos del sistema. (21)

1.5.3 Tipos de Prueba.

Los diferentes tipos de pruebas de software pueden ser agrupados según las dimensiones de calidad a las cuales responden, estas dimensiones están en correspondencia con las características de calidad presentes en la NC ISO/IEC 9126-1:2005. (22)

1.5.3.1 Funcionalidad.

- ✓ Función: Pruebas fijando su atención en la validación de las funciones, métodos, servicios, caso de uso.
- ✓ Seguridad: Asegurar que los datos o el sistema solamente es accedido por los actores deseados y la protección de toda la información contenida en él.
- ✓ Volumen: Enfocada en verificar las habilidades de los programas para manejar grandes cantidades de datos, tanto como entrada, salida o residente en la base de datos.

1.5.3.2 Usabilidad.

- ✓ Usabilidad: Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, documentación de usuarios y materiales de entrenamiento.

1.5.3.3 Fiabilidad.

- ✓ Integridad: Enfocada a la resistencia a fallos del sistema, a su comportamiento ante diversos errores tanto internos como externos.
- ✓ Estructura: Enfocada a la valoración a la adherencia a su diseño y formación. Este tipo de prueba es hecho a las aplicaciones WEB asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.
- ✓ Stress: Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible).

1.5.3.4 Rendimiento (Performance).

- ✓ Benchmark: Compara el rendimiento de un elemento conocido del sistema con un nuevo elemento cuya carga de trabajo se desconoce.
- ✓ Contención: Enfocada a la validación de las habilidades del elemento a probar para manejar aceptablemente la demanda de múltiples actores sobre un mismo recurso (registro de recursos, memoria, etc.).
- ✓ Carga: Usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales.

- ✓ Performance profile: Enfocadas a monitorear el tiempo en flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccional los cuellos de botellas y los procesos ineficientes.

1.5.3.5 Soportabilidad.

- ✓ Configuración: Enfocada a asegurar el funcionamiento en diferentes condiciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema.
- ✓ Instalación: Enfocada a asegurar la instalación en diferentes configuraciones de hardware y software bajo diferentes condiciones.

1.5.4 Métodos de Prueba.

1.5.4.1 Pruebas Basadas en Experiencia.

Son aquellas pruebas que requieren de mayores competencias en el probador y están mayormente enfocadas a las experiencias obtenidas de pruebas anteriores, adivinación de errores y ataques al sistema, son los modos de prueba más comunes dentro de esta técnica. Estas últimas pueden ser realizadas en cualquier situación donde no sea obvio cuál es la próxima prueba que se debe realizar. Puede tener varios objetivos como obtener retroalimentación rápida de cierto producto o funcionalidad, aprender el producto rápidamente, investigar y aislar un defecto en particular, investigar el estado de un riesgo particular, o evaluar la necesidad de diseñar pruebas para determinada área. (21)

1.5.4.2 Pruebas de Caja Blanca.

Las pruebas de Caja Blanca comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código. Mediante las técnicas de prueba de Caja Blanca, el ingeniero de software puede obtener casos de prueba que garanticen que:

- ✓ Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- ✓ Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas.
- ✓ Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez.

La prueba de Caja Blanca, a primera vista, podría parecer impracticable puesto que no es posible aplicarla exhaustivamente para grandes sistemas, sin embargo, no se debe desechar ya que se puede elegir y ejercitar una serie de caminos lógicos importantes, que invoquen además las estructuras de datos más importantes para comprobar su validez (21).

Técnicas de Caja Blanca:

Camino básico: La prueba del camino básico es una técnica de prueba de la Caja Blanca propuesta por Tom McCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.

4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. (21)

Condición: Es un método de diseño de casos prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. Una condición simple es una variable lógica o una expresión relacional. Una condición compuesta está formada por dos o más condiciones simples, operadores lógicos y paréntesis. Por tanto, los tipos posibles de componentes en una condición pueden ser: un operador lógico, una variable lógica, un par de paréntesis lógicos, un operador relacional o una expresión aritmética. (23)

Flujo de datos: Selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa (21)

Bucles: La prueba de bucles se centra exclusivamente en la validez de las construcciones de bucles. Se pueden definir cuatro clases diferentes de bucles: simples, concatenados, anidados y no estructurados (23)

1.5.4.3 Pruebas de Caja Negra.

También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una "Caja Negra" cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. Para seleccionar el conjunto de entradas y salidas sobre las que trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. (23)

Para confeccionar los casos de prueba de Caja Negra existen distintas técnicas. Algunas de ellas son:

Particiones de Equivalencia: Es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Esta se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Las clases de equivalencia se pueden definir según las siguientes directrices:

- ✓Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
- ✓Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
- ✓Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
- ✓Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

Análisis de Valores Límite: El análisis de valores límite (AVL) es una técnica de diseño de casos de prueba que complementa la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.

Métodos Basados en Grafos: El primer paso es entender los objetos que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. Dicho de otra manera, la prueba del software empieza creando un grafo de objetos importantes y sus relaciones, y después diseñando una serie de pruebas que

cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir los errores.

Guiada por Casos de Prueba: Verifican las especificaciones funcionales y no consideran la estructura interna del programa. Se realiza sin el conocimiento interno del producto. No validan funciones ocultas (por ejemplo funciones implementadas pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados.

En otras palabras, la prueba de la Caja Negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden:

- ✓ Demostrar que las funciones del software son operativas.
- ✓ Que las entradas se aceptan de la forma adecuada y que se produce el resultado correcto.
- ✓ Que la integridad de la información externa (por ejemplo archivos de datos) se mantiene.

1.6 Las pruebas de software en el área de las telecomunicaciones.

1.6.1 Tendencia internacional.

En un contexto mundial dominado por grandes compañías que tienen una gran especialización en el desarrollo de software específico para las telecomunicaciones, donde existe una gran competencia entre cada una de estas compañías por obtener una mayor porción del creciente mercado que estos sistemas tienen en el mundo. (24) En consulta realizada a los sitios oficiales de las diferentes empresas líderes en el desarrollo de software para las telecomunicaciones, tales como AT&T, Motorola, Teltronic, CISCO, Telefónica, Ericcson, se evidenció que ninguna comparte información referente al proceso de desarrollo de sus productos, con el fin de buscar una metodología propia y cada vez más eficiente que les permita adelantarse siempre a la competencia con productos superiores en rendimiento y con menos defectos. Esta dinámica de

mercado ha hecho que las compañías mejoren sus estrategias de pruebas de software para lograr sistemas sin errores y con menor tiempo en su ciclo de desarrollo, pero a su vez ha convertido en propiedad de cada una de las multinacionales la información referente a las pruebas, razón por la cual es importante desarrollar una estrategia de pruebas específica para el Centro de Telemática.

1.6.2 Proceso de pruebas en el Departamento de Soluciones Informáticas en ETECSA.

En el país la entidad rectora de las telecomunicaciones es la Empresa de Telecomunicaciones de Cuba S.A. (ETECSA), la cual se encarga de ofrecer los diferentes servicios asociados. Esta empresa se encuentra dividida en vicepresidencias entre las cuales está la Vicepresidencia de Tecnologías de la Información (VPTI). En visita realizada al Departamento de Soluciones Informáticas ubicado en el Centro de Negocios Miramar, edificio Jerusalén, se pudo conocer que para realizar las pruebas de software cuentan con el Grupo de Implementación y Pruebas, dicha tarea en estos momentos la hace una sola ingeniera y no cuentan con una estrategia enfocada a las características específicas de los sistemas de telecomunicaciones, como puede verse en el Plan de Pruebas que en esa entidad es usado (Anexo 2). Debido a la falta de personal dedicado a la realización de pruebas y a la cantidad de proyecto asumidos por el departamento, en la práctica esa estrategia de pruebas no puede cumplirse. Dentro del estudio realizado se detectó que sólo se realizan pruebas de Caja Negra sobre el sistema, en el caso de las Pruebas de Seguridad son realizadas por el departamento de seguridad informática de ETECSA que es quien da la aprobación para la puesta en explotación de cualquier sistema informático de la entidad. Entre los mecanismos de pruebas dentro de la técnica de Caja Negra, se utilizan las Pruebas de Particiones de Equivalencia, Análisis de Valores Límites, Análisis Causa Efecto y Guiada por Casos de Prueba. Entre las técnicas usadas se encuentra la Prueba Basada en Experiencias. También se pudo conocer que

todas las pruebas se realizan de manera manual, por lo cual no se usa ninguna de las herramientas disponibles para llevar a cabo esta actividad.

1.6.3 Proceso de pruebas en la UCI.

En la universidad, específicamente en el Centro de Telemática de la Facultad 2, no se lleva a cabo una estrategia de pruebas adecuada para este tipo de sistemas y solo se realizan pruebas generales aplicables a todo tipo de software, dichas pruebas son realizadas por el Grupo de Calidad de la Facultad y Calisoft. En entrevista realizada a integrantes de proyectos productivos pertenecientes al Centro, se preguntó acerca de las técnicas de prueba que se ponen en práctica; resultó que entre los métodos de Caja Negra, la Prueba Guiada por Casos de Prueba es la que se pone en práctica, y entre los de Caja Blanca, la Prueba del Camino Básico. Todo lo anterior demuestra que no se realiza de manera adecuada el proceso de pruebas específicamente para este tipo de sistemas.

1.7 Conclusiones Parciales.

En este capítulo se trataron los conceptos relacionados con las pruebas de software. Se analizó la NC ISO/IEC 9126:2005 de la cual solamente se profundizó en el primero de sus acápites. Se analizaron las entrevistas realizadas a especialistas en el desarrollo de software para las telecomunicaciones para saber cuáles de las características son las más relevantes en este software. Se estudiaron las metodologías de desarrollo de software utilizadas en el Centro de Telemática y se dio a conocer la situación de las pruebas de software en el área de las telecomunicaciones.

Capítulo 2: Propuesta de la Estrategia de Pruebas.

Introducción

El objetivo de este capítulo es describir la estrategia de prueba de software propuesta para el Centro de Telemática de la Facultad 2, la cual tributa a las características específicas del software para las telecomunicaciones. También se realiza un análisis de las metodologías de desarrollo utilizadas en el Centro, identificando actividades genéricas durante el proceso de pruebas, los artefactos utilizados para llevar a cabo las mismas y sus responsables. Finalmente, se proponen las herramientas utilizar para la automatización del proceso de prueba de software.

2.1 Descripción de las Actividades.

Luego del estudio de las distintas metodologías usadas en el Centro, se identificaron una serie de actividades que, independientemente de la metodología utilizada, son genéricas para desarrollar un satisfactorio proceso de pruebas. La estrategia de pruebas tributa directamente a la actividad de Diseñar Pruebas, aunque de manera indirecta también lo hace en las restantes actividades, debido a que ella es la rectora del proceso de pruebas.

2.1.1 Planificar Pruebas.

En esta actividad se define el alcance de las pruebas así como sus objetivos, además de tener en cuenta los recursos humanos y materiales necesarios para su realización. Este plan puede variar en toda la vida del proyecto, por lo cual dentro de la planificación, debe contemplarse la redefinición del mismo. En los proyectos cuya metodología de desarrollo sea RUP el Administrador de Pruebas es el encargado de realizar esta actividad. En los que usen sXP, el Analista y el Encargado de Pruebas serán las personas responsables de esta labor. En los proyectos donde sea FDD la metodología usada, el encargado de esta actividad debe ser el *Tester* o Probador, de conjunto con el *Technical Writer* o Escritor de documentos técnicos.

2.1.2 Diseñar Pruebas.

Esta actividad tiene como objetivos fundamentales identificar, describir y diseñar los Casos de Prueba. Luego de identificadas las pruebas, se procede a seleccionar las técnicas que se deben utilizar para el tipo de software con el que se trabaja, analizar los objetivos de las pruebas que se han planificado y diseñar los casos de prueba. RUP define que esta labor debe ser realizada por el Diseñador de pruebas. Para el caso de sXP el Encargado de Pruebas será quien la ejecute. En los proyectos que trabajen bajo la metodología FDD el Probador será el responsable de realizar esta labor.

2.1.3 Configurar Ambiente de Pruebas.

El propósito de esta actividad es asegurar todas las condiciones de software y hardware que sean necesarias para la ejecución de las pruebas, además de garantizar las herramientas adecuadas y la capacitación del equipo de probadores seleccionado previamente. También se incluye la actividad de implementar los Casos de Prueba que se han definido durante el diseño, automatizando todos los que sean posibles. En la metodología RUP el encargado de realizar esta actividad es el Ingeniero de componentes, aunque pueden incorporarse el Diseñador y el Probador. Para la metodología sXP esta responsabilidad recae sobre el Encargado de Pruebas. En FDD se gestiona esta actividad entre el Probador, el Herramentista y el Administrador de Sistema.

2.1.4 Ejecutar las Pruebas.

El objetivo de esta actividad es realizar las Pruebas de Unidad, Integración y Sistema. Durante esta actividad se ejecutan las pruebas previamente planificadas y diseñadas. El encargado de ejecutar todas las pruebas es el Probador para la metodología RUP, y el Encargado de Pruebas tanto en sXP como en FDD. Durante este proceso se realizan reportes de No Conformidades (NC), utilizando la plantilla propuesta por la Dirección de Calidad de la Universidad en la última versión del expediente de proyecto que puede ser consultada en el sitio de Calisoft, y se registran los resultados.

2.1.5 Concluir Pruebas.

En esta actividad se realiza un análisis final de las pruebas aplicadas y se confeccionan informes finales. Bajo la metodología RUP los Diseñadores de pruebas llevan a cabo esta actividad revisando y evaluando los resultados de las pruebas. Luego el Administrador de Pruebas, se encarga de emitir el documento donde se registran las NC finales, utilizando la plantilla propuesta por la Dirección de Calidad de la Universidad en la última versión del expediente de proyecto. Para las metodologías sXP y FDD el Encargado de Pruebas es el responsable de llevar a cabo esta actividad.

2.2 Descripción de los Artefactos.

2.2.1 Plan de Pruebas.

Este plan tiene como propósito definir el alcance, enfoque, recursos requeridos, calendario, responsables, organización y estrategia de las pruebas. El mismo se realizará haciendo uso de la plantilla propuesta por la Dirección de Calidad de Software de la Universidad en la versión 2.02 del expediente de proyecto.

2.2.2 Expediente de Pruebas.

La intención de este artefacto es archivar todos los documentos generados durante la etapa de pruebas. El expediente de pruebas debe incluir:

1. Diseño de Casos de pruebas.

2. Especificación de Casos de pruebas.

3. Reporte de No Conformidades.

4. Resumen de la Jornada de trabajo.

5. Componentes automatizados.

Todos estos documentos se deben redactar siguiendo las plantillas propuestas y se deben recoger en una carpeta con el nombre de Expediente de Pruebas.

2.2.3 Casos de Prueba.

Este artefacto tiene como propósito definir un conjunto de condiciones o variables bajo las cuales se determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Los Casos de Pruebas se deben diseñar a partir de la plantilla propuesta por la Dirección de Calidad de la Universidad en la última versión del expediente de proyecto.

2.2.4 Reporte de No Conformidades (NC).

En este documento se deben recoger las NC detectadas en el proceso de ejecución de los distintos Casos de Prueba. Los probadores son los responsables de realizar esta actividad, la cual debe ser desarrollada utilizando la plantilla propuesta por la Dirección de Calidad de la Universidad en la última versión del expediente de proyecto.

2.2.5 Reporte Final de NC.

Es el documento final que recoge todas las NC detectadas a lo largo de la ejecución de las pruebas. Este debe ser desarrollado a partir de la plantilla propuesta por la Dirección de Calidad de la Universidad.

2.2.6 Resumen de la Jornada de Trabajo.

En este documento se guardan los resultados de toda la jornada de trabajo y se crea un resumen de las NC detectadas en el transcurso del día.

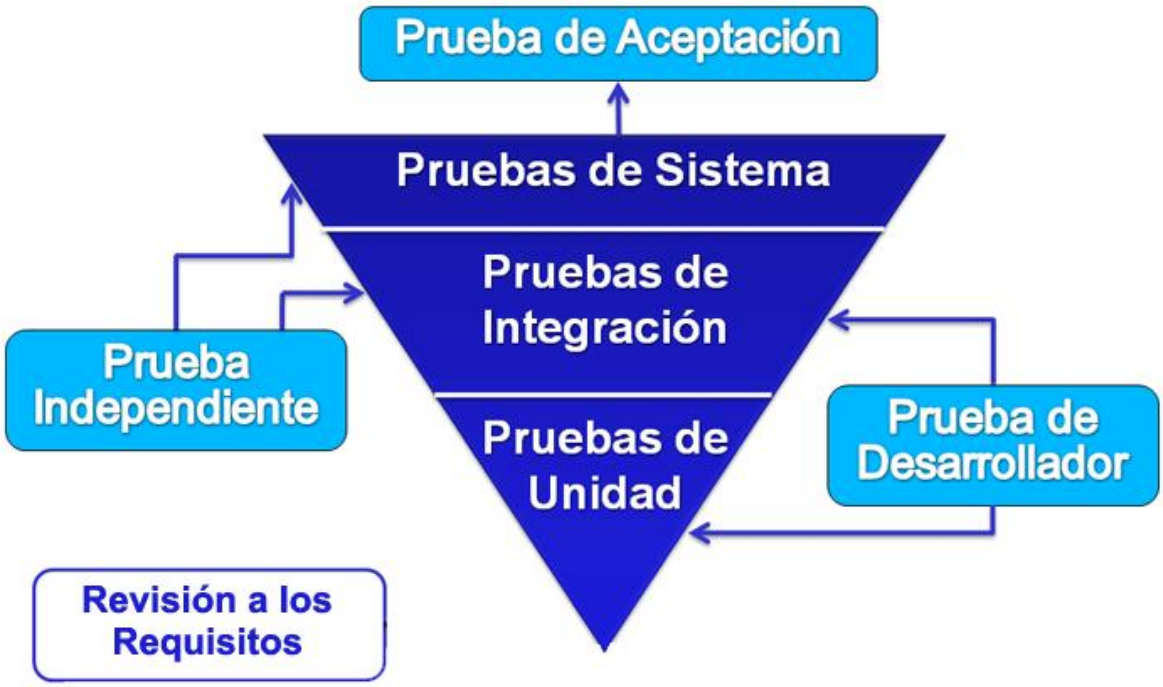
2.3 Definición de la Estrategia.

Según la investigación llevada a cabo se determinó que por las características que presenta el software para las telecomunicaciones, se liberará por parte del grupo de calidad de la facultad los requisitos, debido a la vital importancia dada a estos por los especialistas entrevistados. Luego se propone una estrategia desde

adentro hacia afuera, partiendo de las pruebas de unidad para llegar a las pruebas de sistema y aceptación. En los proyectos cuya metodología de desarrollo sea RUP se propone que las pruebas de integración sean realizadas por personas ajenas al equipo de desarrollo, y para los que utilicen sXP o FDD las pruebas de integración serán pruebas de desarrollador. Los tipos de prueba definidos en cada nivel se corresponden con las dimensiones de calidad encaminadas a las características de la NC ISO/IEC 9126-1:2005.

Para el seguimiento y control de las No Conformidades, debido a que la universidad está inmersa en un proceso de mejoras para optar por el nivel 2 de CMMI, el aseguramiento y control de la calidad debe regirse por lo definido en el área de Aseguramiento de la Calidad del Proceso y el Producto (PPQA del inglés Process and Product Quality Assurance), está propone para el seguimiento de las NC detectadas un subproceso del cual sus actividades están enfocadas en llevar un seguimiento adecuado mediante el documento de acciones correctivas.

Figura 2.1: Estrategia de Prueba.



2.3.1 Revisión de los Requisitos.

Se dice que la base para un software con calidad es un buen levantamiento de requisitos, debido a la vital importancia de este proceso se recomienda que el documento de Especificación de Requisitos sea liberado por el grupo de calidad del Centro, utilizando la plantilla propuesta por la Dirección de Calidad de la Universidad en la última versión del expediente de proyecto.

2.3.2 Pruebas de Unidad.

Las pruebas a nivel de Unidad son realizadas para comprobar el funcionamiento de las clases que conforman el proyecto, dentro de este nivel se proponen realizar pruebas de función y seguridad. Utilizando para ello el método de prueba de Caja Blanca, y dentro de sus técnicas la del camino básico.

2.3.3 Pruebas de Integración.

Es el nivel de prueba donde se garantiza una adecuada integración entre los módulos o componentes del sistema. Dentro de este se propone la realización de pruebas específicas de función, seguridad e integridad. Estas pruebas deben ser realizadas usando el método de Caja Negra, empleando para ello las técnicas guiadas por casos de prueba y el método basado en grafos. En las pruebas de integración se deben usar basadas en hilos, se deben integrar las clases necesarias para responder a un evento del sistema, cada hilo se integra y prueba individualmente.

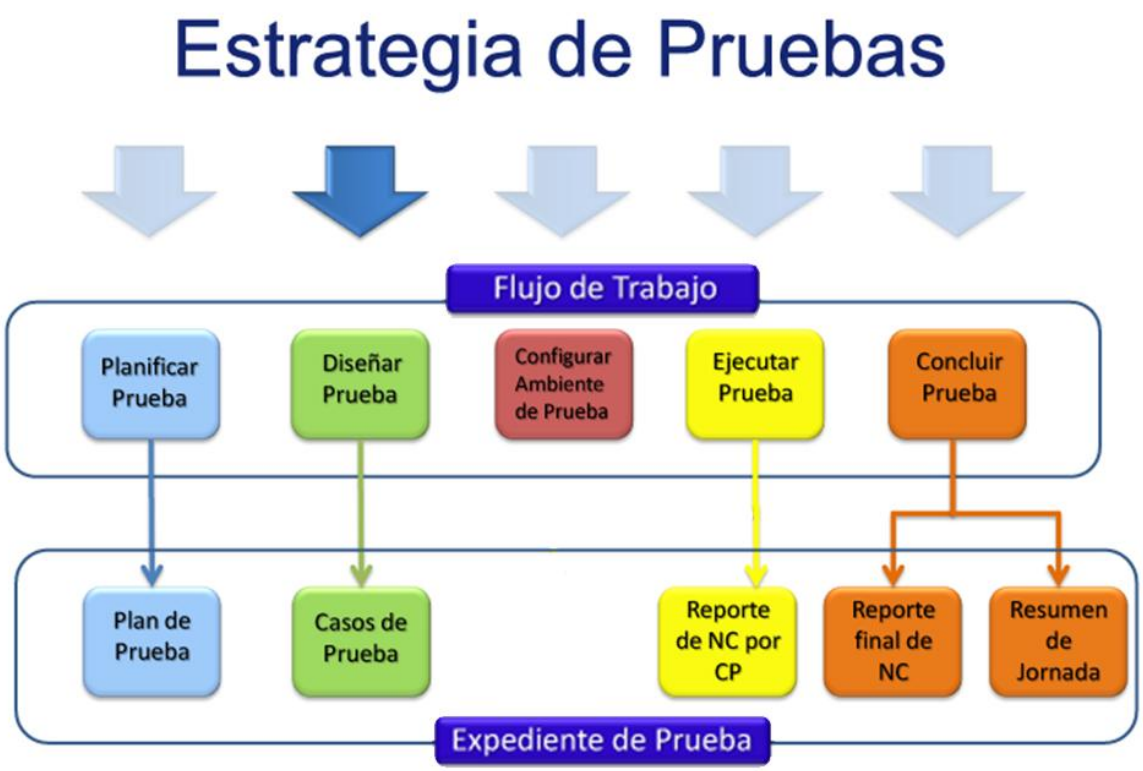
2.3.4 Pruebas de Sistema.

En este nivel de prueba se comprueba el sistema trabajando como un todo y se verifica que cumpla en su conjunto todas las funciones para las que fue concebido, se proponen pruebas de funcionalidad, seguridad, rendimiento, fiabilidad y soportabilidad. Utilizando para esto el método de pruebas de Caja Negra.

2.3.5 Pruebas de Aceptación.

Estas serían las pruebas realizadas por el cliente donde se deben revisar cada uno de los requisitos funcionales del sistema. Para este nivel de pruebas se propone utilizar pruebas encaminadas a la funcionalidad, a la fiabilidad, a la seguridad y al rendimiento.

Figura 2.2: La Estrategia de Pruebas dentro del Flujo de Trabajo de Pruebas.



2.4 Descripción de la Estrategia.

En este acápite se describen los tipos de prueba a aplicar en cada uno de los niveles, así como la forma de aplicarlas y sus responsables. Para mejor entendimiento de la estrategia propuesta la descripción de cada una de las pruebas constará de los siguientes aspectos:

Descripción: [Descripción general de la prueba a aplicar].

Objetivo: [Objetivo que persigue la prueba a aplicar].

Técnica: [Descripción detallada de la manera en la que se prueba].

Responsable de ejecución: [Responsable de ejecutar la prueba].

Responsable de diseño: [Responsable de diseñar la prueba].

Para cada una de las pruebas se crea un expediente que incluye todo lo que se genere durante el proceso de pruebas. Una vez diseñados y aprobados los Casos de Prueba el encargado de realizar las pruebas los ejecuta y documenta los resultados obtenidos, si se detecta alguna no conformidad, estas deben ser registradas en el reporte de No Conformidades.

2.4.1 Pruebas de Unidad.

En el nivel de Unidad se van a realizar pruebas centradas en comprobar la funcionalidad del sistema. Las pruebas de Función y de Seguridad son las seleccionadas para ser llevadas a cabo en este nivel, donde las pruebas serán realizadas por los propios programadores como se muestra en la figura a continuación.

Fig. 2.3: Pruebas a llevar a cabo en el nivel de unidad.

2.4.1.1 Pruebas encaminadas a la Funcionalidad.

- ✓ Prueba de Función.

Descripción: Es una prueba centrada en validar las funciones que son objeto de prueba, se aplican para verificar que la unidad a ser probada se adecua a los requisitos funcionales. Al realizar esta prueba se debe comprobar la idoneidad del sistema debido a que el software debe ser capaz de realizar las tareas específicas y los objetivos del usuario. También la precisión debe ser comprobada en esta prueba para verificar la capacidad del sistema de propiciar efectos o resultados correctos.

Objetivo: Se garantiza la funcionalidad del sistema comprobando mediante el recorrido de los distintos caminos posibles en el código que este maneja adecuadamente los datos y realiza las operaciones necesarias.

Técnica: La técnica la del Camino Básico perteneciente al método de Caja Blanca.

Responsable de ejecución: Programador.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD la prueba debe ser diseñada por el Propietario de la Clase y en el caso de sXP por el propio Programador.

- ✓ Prueba de Seguridad.

Descripción: La prueba de seguridad dentro del nivel de unidad estará encaminada a garantizar una programación segura mediante análisis estático del código. Cuando se habla de código seguro se entiende por código diseñado para soportar ataques de usuarios maliciosos y ser capaz de detenerlos. La Programación Segura como rama de la Seguridad Informática y la programación, define patrones que deben ser seguidos por los desarrolladores para lograr sistemas robustos, fiables y libres de errores, la misma hace uso de técnicas de pruebas de software, utilización de funciones seguras, control del flujo de la información de un programa, entre otras, las que propician estabilidad y certeza del código escrito. Los principales objetivos de la seguridad son la autenticidad, la confidencialidad, la integridad y la disponibilidad. Uno de los lenguajes más usados en las aplicaciones para las telecomunicaciones es Java, el cual se considera seguro debido a que el verificador embebido, *(bytecode verifier)* asegura que no haya explícita manipulación de punteros y de arreglos, que las cadenas de caracteres se comprueben automáticamente, que se capturen los intentos de referenciar a un puntero nulo y que las operaciones aritméticas y de conversión de tipos estén bien definidas e independientes de plataforma. Además, en Java existen unos buenos mecanismos de seguridad que controlan el acceso a ficheros individuales, *sockets* y otros recursos sensibles. Aún así las aplicaciones Java están expuestas a diferentes ataques, por tanto, se debe comprobar en ellas su seguridad.

Objetivo: Garantizando una programación segura, es decir, reduciendo al mínimo sus vulnerabilidades que es su objetivo, se logra aumentar la seguridad del sistema en el momento que este se encuentre trabajando como un todo y se previenen ataques que comprometan la integridad del software.

Técnica: La técnica la del Camino Básico perteneciente al método de Caja Blanca.

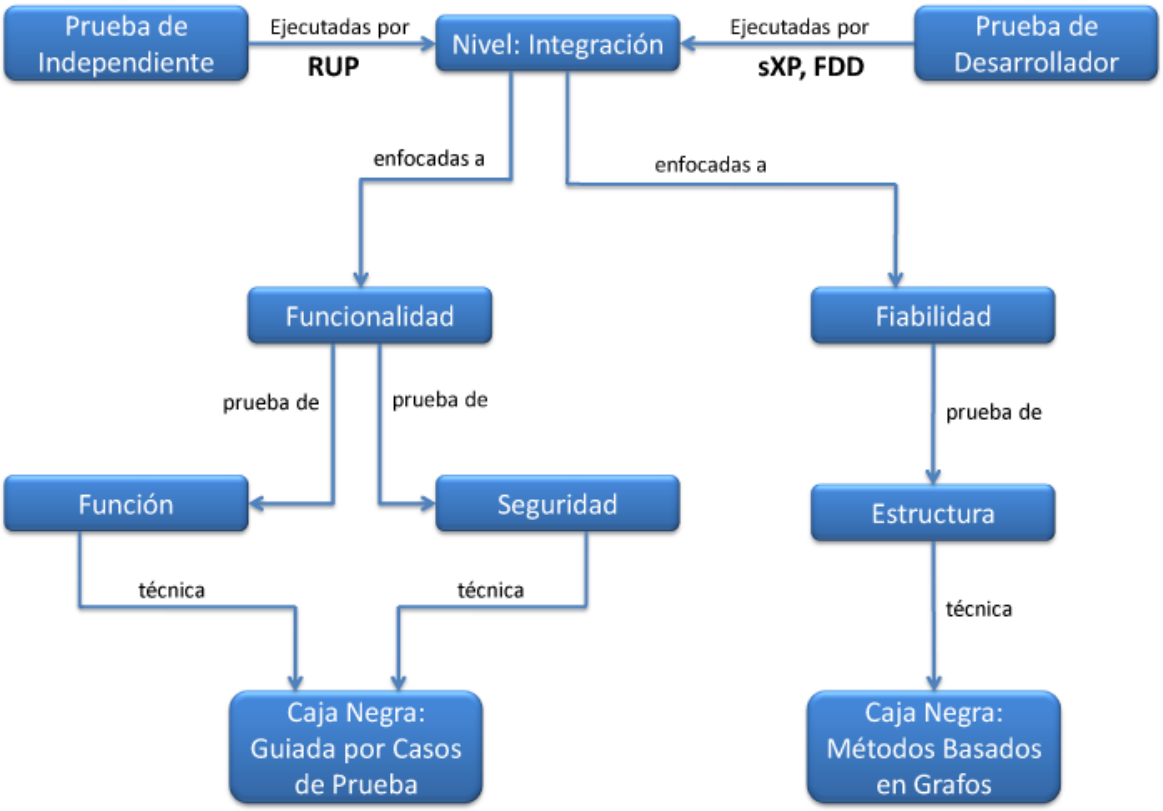
Responsable de ejecución: Programador.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD la prueba debe ser diseñada por el Propietario de la clase y en el caso de sXP el Programador será el encargado de diseñar esta prueba.

2.4.2 Pruebas de Integración.

En el nivel de integración se llevaran a cabo pruebas enfocadas a la funcionalidad, al igual que en el nivel anterior. Además de estas se realizara la prueba de estructura dirigida a comprobar la fiabilidad del producto. Este nivel presenta la particularidad de que el encargado de realizar las pruebas está en función de la metodología de desarrollo utilizada por el proyecto que produce el software. En la figura 2.4 se muestra de forma breve la organización de las pruebas en este nivel.

Fig. 2.4: Pruebas a llevar a cabo en el nivel de integración.



2.4.2.1 Pruebas encaminadas a la Funcionalidad.

- ✓ Prueba de Función

Descripción: En esta prueba se debe comprobar la funcionalidad de las partes del sistema que depende de otras para su funcionamiento. Estas pruebas se realizaran de manera ascendente, desde el momento en que dos módulos o partes del sistema que dependan una de la otra para trabajar se hayan implementado, hasta la mayor cantidad de módulos del sistema que necesiten trabajar de manera conjunta.

Objetivo: Usando el mecanismo descrito se comprobara la funcionalidad del sistema verificando el correcto manejo de los datos y su apropiada respuesta mientras el mismo se va integrando.

Técnica: La técnica a utilizar será la Guiada por Casos de Prueba dentro del método de Caja Negra. Se debe integrar de forma ascendente, ir combinando los módulos de los niveles inferiores moviéndose hacia los superiores por la estructura del programa.

Responsable de ejecución: Probadores del grupo de calidad del centro en el caso de RUP. En el caso de los proyectos que trabajen con sXP o FDD estos serán ejecutados por los propios Programadores

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el encargado de pruebas.

Es aconsejable que paralelamente al proceso de integración se apliquen las pruebas de regresión, con el objetivo de asegurar que las fallas detectadas en los módulos sean corregidas y que no se introduzcan nuevos errores al tratar de solucionar los detectados anteriormente. Las pruebas de regresión consisten en aplicar el mismo proceso de pruebas planificado originalmente, sólo que se repiten cada vez que se corrige un resultado erróneo.

- ✓ Prueba de Seguridad

Descripción: Esta prueba se ejecuta para comprobar que las partes del sistema funcionando de manera conjunta sean capaces de garantizar la seguridad de los datos que este maneja. Se debe comprobar la seguridad de forma ascendente, ir combinando los módulos de los niveles inferiores moviéndose hacia los superiores por la estructura del programa. En el caso de sistema de base de datos se deberá de intentar la inyección de código SQL.

Objetivo: Garantizar la seguridad del software a la medida que sus diferentes componentes se van integrando intentado acceder de manera no autorizada a la información del sistema. De igual modo se busca con esta prueba comprobar que cada usuario solo tenga acceso al sistema una vez autenticado y lo haga únicamente a los lugares autorizados.

Técnica: El método a utilizar para este tipo de pruebas será la prueba de caja negra utilizando la técnica guiada por casos de prueba para tratar de acceder a los datos del sistema y modificarlos o eliminarlos.

Responsable de ejecución: Probadores de grupo de calidad del centro en el caso de RUP. En el caso de los proyectos que trabajen con sXP o FDD estas serán ejecutadas por los programadores.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el encargado de pruebas.

2.4.2.2 Pruebas encaminadas a la Fiabilidad.

- ✓ Prueba de Estructura

Descripción: Esta prueba está enfocada a la valoración a la adherencia a su diseño. En aplicaciones web este tipo de prueba es hecho asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano. En esta prueba se debe comprobar el mapa de navegación de la aplicación.

Objetivo: Esta prueba pretende comprobar la fiabilidad del sistema verificando la correcta conexión entre cada uno de sus módulos o secciones. Haciendo uso del mecanismo propuesto se deben recorrer cada uno de los caminos dentro de la aplicación verificando que no existan callejones sin salida.

Técnica: La técnica a utilizar será Basada en Grafo dentro del método de Caja Negra.

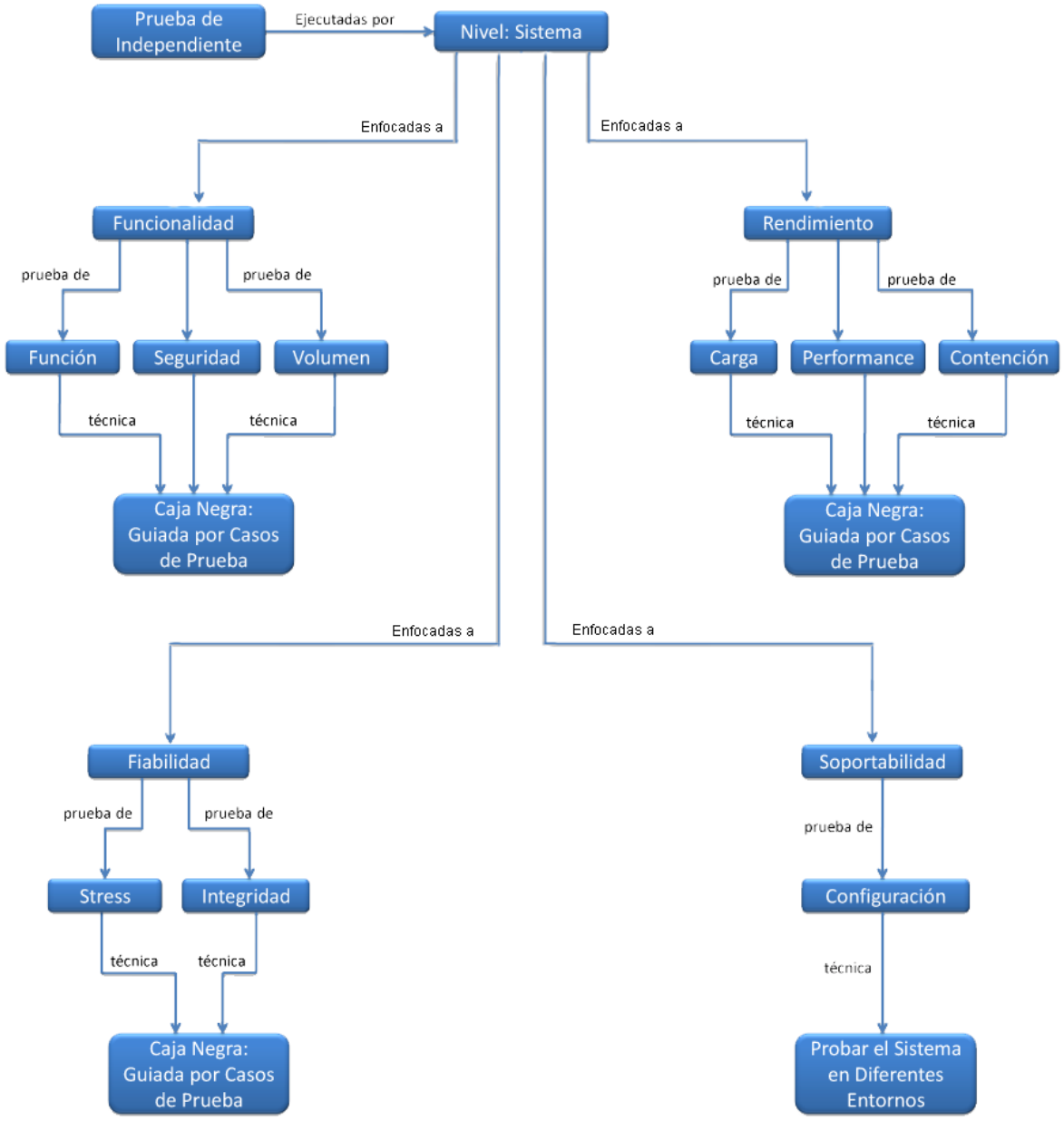
Responsable de ejecución: Probadores del grupo de calidad del centro en el caso de RUP. En el caso de los proyectos que trabajen con sXP o FDD estos serán ejecutados por los programadores.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o XP debe ser diseñada por el encargado de pruebas.

2.4.3 Pruebas de Sistema.

En este nivel de prueba se debe comprobar el sistema trabajando como un todo. Se deben revisar sus características funcionales y no funcionales así como su capacidad para trabajar en el ambiente establecido por el cliente. Estas pruebas serán ejecutadas por personas ajenas al equipo de desarrollo. Las características de calidad que en este nivel se comprueban, los tipos de pruebas propuestas y las técnicas aplicadas puede observarse en la figura 2.5.

Fig. 2.5: Pruebas a llevar a cabo en el nivel de sistema.



2.4.3.1 Pruebas encaminadas a la Funcionalidad.

- ✓ Prueba de Función

Descripción: Esta prueba es llevada a cabo para garantizar que el sistema cumpla con los requisitos funcionales que fueron especificados con el cliente es decir que haga las funciones para las cuales fue diseñado

Objetivo: Los requisitos funcionales del sistema serán comprobados haciendo uso de juegos de datos para los cuales se espera una salida que sea la adecuada, para así asegurar que el software realmente cumple con todos sus requisitos funcionales.

Técnica: Se empleará la técnica basada en casos de prueba perteneciente al método de Caja Negra, entrándole al sistema juegos de datos y comprobando que los devueltos sean los requeridos.

Responsable de ejecución: Las pruebas serán ejecutadas por los Probadores del grupo de calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el encargado de pruebas.

- ✓ Prueba de Seguridad

Debido a que la mayor parte de las aplicaciones que se desarrollan en el centro y en general la tendencia mundial de la industria del software es el uso de tecnologías web para realizar estas pruebas se deben tener en cuenta los 5 problemas fundamentales que según OWASP (Open *Web* Application Security Project) (25), son:

- ✓ Inyección: ocurre cuando existen datos inseguros que se envían a un intérprete como parte de un comando o consulta;
- ✓ Cross site scripting (*XSS*): ocurre cuando una aplicación toma datos no seguros y los envía a un navegador *Web* sin la validación y escape necesarios, lo que trae como consecuencia la ejecución de scripts en el navegador de la víctima que producen daños terribles para el cliente.
- ✓ Autenticación y administración de sesión destruida: se relaciona con la autenticación y administración de sesiones, las cuales no son correctamente implementadas y permiten a los atacantes comprometer contraseñas, llaves, etc., todo con el objetivo de asumir identidades falsas.

- ✓ Referencias inseguras de objetos directos: se producen cuando un desarrollador expone una referencia de una implementación interna de un objeto como un fichero, directorio o llave de una Base de Datos, lo cual provoca que los atacantes manipulen las referencias para acceder a datos no autorizados.
- ✓ Cross site request forgery (CSRF): se fuerza al navegador *Web* validado de una víctima a enviar una petición a una aplicación *Web* vulnerable, la cual entonces realiza la acción elegida a través de la víctima. Mediante CSRF se explota la confianza que un sitio tiene en un usuario en particular.

Descripción: Esta prueba se ejecuta para comprobar que el sistema funcionando como un todo sea capaz de garantizar la seguridad de los datos que este maneja.

Objetivo: La seguridad del sistema debe ser revisada intentando acceder de manera no autorizada o realizar cambios en los datos del sistema sin previa autenticación para ello.

Técnica: Se empleará la técnica Basada en Casos de Prueba dentro del método de Caja Negra.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los Probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el Encargado de pruebas.

- ✓ Prueba de Volumen

Descripción: Esta prueba se ejecuta para comprobar el funcionamiento del sistema ante grandes volúmenes de información presentes en los datos con que trabaja.

Objetivo: El fin de esta prueba es comprobar la funcionalidad del sistema con grandes cantidades de información

Técnica: Se empleará la técnica Basada en Casos de Prueba dentro del método de Caja Negra, usando para ello la herramienta Push To Test Test Maker.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los Probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el encargado de pruebas.

2.4.3.2 Pruebas encaminadas al Rendimiento.

- ✓ Prueba de Carga

Descripción: Es usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable. Para la carga de trabajo a que va a ser sometido el software se deben tener en cuenta los picos reales de carga que debe afrontar cuando sea entregado al cliente y se encuentre en su ambiente real de trabajo.

Objetivo: El rendimiento del sistema es medido en esta prueba mediante su trabajo ante distintas condiciones de trabajo, la herramienta propuesta posibilita monitorear la respuesta del sistema ante distintas condiciones.

Técnica: Se empleará la técnica Basada en Casos de Prueba dentro del método de Caja Negra.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los Probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el encargado de pruebas.

- ✓ Prueba de Performance Profile

Descripción: Esta prueba se ejecuta para verificar el tiempo de reacción de sistema ante distintas situaciones que se enfrentara en su puesta en marcha.

Objetivo: Mediante la comprobación y el seguimiento de los tiempos de respuesta del sistema se asegura su adecuado rendimiento una vez puesto en marcha bajo su régimen habitual de trabajo.

Técnica: Se empleará la técnica Basada en Casos de Prueba dentro del método de Caja Negra, utilizando la herramienta propuesta para esto.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los Probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el Encargado de Pruebas.

- ✓ Prueba de Contención

Descripción: Esta prueba se ejecuta para comprobar el funcionamiento del software cuando un mismo recurso es solicitado por varios usuarios a la vez.

Objetivo: El rendimiento del sistema es medido en esta prueba utilizando la herramienta propuesta con el fin de comprobar la adecuada reacción del sistema ante él un múltiple pedido sobre el mismo recurso.

Técnica: Se empleará la técnica Basada en Casos de Prueba dentro del método de Caja Negra.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los Probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el Encargado de Pruebas.

Procedimiento para realizar la prueba: Se usara la herramienta propuesta para las pruebas de sistema. Esta nos permite hacer peticiones simultáneas sobre un mismo recurso por parte de varios usuarios.

2.4.3.3 Pruebas encaminadas a la Fiabilidad.

- ✓ Prueba de Stress

Descripción: Está enfocada a evaluar cómo el sistema responde bajo condiciones anormales de trabajo. Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible.

Objetivo: Comprobar la fiabilidad del sistema para enfrentarse durante las funciones para las cuales fue diseñado a condiciones anormales en su entorno de trabajo.

Técnica: Se empleará la técnica Basada en Casos de Prueba dentro del método de Caja Negra.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los Probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el Encargado de Pruebas.

- ✓ Prueba de Integridad

Descripción: Esta prueba se ejecuta para comprobar el funcionamiento del sistema y su recuperación ante fallos provocados por errores del propio sistema o por otros de software o hardware que interactúen con él.

Objetivo: Comprobar la confiabilidad del sistema y su recuperación ante posibles errores. Se deben provocar fallos controlados en el sistema y en otros asociados a él, de los cuales dependa para realizar su labor, comprobar cómo responde ante cada uno de estos problemas y hasta qué punto es capaz de seguir brindando el servicio que de él se espera.

Técnica: Se empleará la técnica Basada en Casos de Prueba dentro del método de Caja Negra. Los casos de prueba deben ser diseñados teniendo en cuenta todos los posibles fallos en el entorno de trabajo real del sistema.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los Probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el Encargado de Pruebas.

2.4.3.4 Pruebas encaminadas a la Soportabilidad.

- ✓ Prueba de Configuración

Descripción: Esta prueba se ejecuta para comprobar la adaptabilidad del sistema ante distintos entornos de hardware y software.

Objetivo: Verificar que el sistema sea soportado por una gran variedad de hardware y pueda trabajar asociado a la mayor cantidad posible de software interferir con su labor.

Técnica: Para realizar esta prueba se deberá utilizar el sistema en una gran cantidad de configuraciones de hardware así como asociados a distintos software para comprobar su capacidad de interactuar sin crear conflictos con otros sistemas y configuraciones.

Responsable de ejecución: Los responsables de realizar dicha prueba serán los probadores del Grupo de Calidad del centro.

Responsable de diseño: Diseñador de Pruebas en el caso de RUP, en los casos que el proyecto trabaje con FDD o sXP debe ser diseñada por el Encargado de Pruebas.

2.4.4 Pruebas de Aceptación.

Estas son las pruebas finales de aceptación del producto por parte del cliente, en esta comprueba que el sistema cumpla cada uno de los requisitos funcionales y no funcionales que fueron especificados. El responsable de diseñar y ejecutar esta prueba es el cliente y en ella debe verificar la conformidad del sistema con lo

planteado en los documentos firmados por este. Estas pruebas deben realizarse después de concluidas las de sistema.

2.5 Herramientas de Software Propuestas.

En la actualidad existen una serie de herramientas que pueden ayudar en la automatización de los procesos de pruebas de software, estas no deben verse como la parte principal de las pruebas de software, es complementaria y ayuda a una mayor eficiencia en su realización. Para una buena automatización se requiere un buen diseño de casos de prueba. Y se debe prever el costo y los recursos necesarios para el proceso de automatización.

2.5.1 Herramientas en el nivel de Unidad.

Para proponer una herramienta adecuada con el fin de realizar las pruebas en el nivel de unidad se tiene en cuenta que sean herramientas de código abierto y con la madurez requerida para realizar estas pruebas, existen dos software que cumplen con las características antes mencionadas, estos son JUnit y TestNG.

JUnit fue diseñado expresamente para pruebas de unidad en lenguaje Java. Es la herramienta más usada por los desarrolladores que trabajan este lenguaje para realizar sus pruebas en este nivel.

Test Next Generation (TestNG) es una herramienta que aun siendo diseñada para pruebas de unidad es capaz de realizar pruebas integración, tiempo de algoritmos y otras. Fue creada para superar las principales limitaciones de JUnit y proporciona características adicionales necesarias para probar la última generación de aplicaciones en Java. Otra de sus ventajas es el agrupamiento de las pruebas, característica que facilita su uso por parte del usuario. Permite que los métodos a ser probados puedan contar con parámetros, estos pueden ser obtenidos de forma dinámica a través de base de datos u otro archivo de almacenamiento. Esta herramienta cuenta con plugins en los IDE más usados como son Eclipse, NetBeans y Visual Studio. (26)

Debido a las ventajas mencionadas se propone el uso de esta herramienta para realizar las pruebas de función en el nivel de unidad.

Para realizar las pruebas de seguridad en este nivel, se propone el uso de herramientas adecuadas al lenguaje de programación usado en el desarrollo del proyecto. Para los sistemas desarrollados utilizando los lenguajes PHP y Phyton, se propone utilizar las herramientas desarrolladas con este objetivo por el laboratorio de seguridad de la facultad. Estas se nombran Análisis Estático de Vulnerabilidades en PHP y Phyton respectivamente y brindan la posibilidad de reducir desde el código la posibilidad de ataques con el sistema.

Para el lenguaje Java, el laboratorio de seguridad no cuenta aún con una herramienta destinada a él. Por lo cual se propone utilizar el FindBugs, una herramienta de código abierto desarrollada en la Universidad de Maryland. Este sistema trabaja en Java *Bytecode* en lugar de hacerlo en código fuente lo cual posibilita encontrar en número mayor de errores.

FindBugs clasifica los errores en las siguientes categorías:

- ✓ Correcto pero probable error: un error de codificación aparente resultando en código que probablemente no era lo que el programador tenía en mente.
- ✓ Mala Prácticas: violaciones a las prácticas de codificación esenciales y recomendadas, por ejemplo: uso incorrecto del *finalize*.
- ✓ Código Dudoso: esto es código confuso, anómalo, o escrito de una manera que lleva a sí mismo a errores, por ejemplo: *casts* sin confirmar, chequeos de nulos redundantes, etc.

FindBugs puede ser utilizado de conjunto con los IDE de desarrollo más utilizados para el lenguaje Java como Eclipse y Netbeans (27).

2.5.2 Herramientas para las Pruebas de Sistema.

Para las pruebas en el nivel de sistema durante la investigación se encontraron 2 herramientas que cumplen los requisitos de ser libres y presentar una robustez

adecuada para su uso generalizado por el centro, estas fueron Jmeter y PushToTesT TestMaker.

Jmeter es una herramienta creada 100 % en java, entre sus funcionalidades nos permite realizar pruebas de carga, stress y volumen sobre aplicaciones web. Además, se pueden realizar test a servidores FTP, JBDC, JNDI, LDAP, SOAP/XML-RPC. Jmeter puede también realizar pruebas de regresión a los diferentes sistemas y posee una gran variedad de informes con los resultados. (26)

PushToTest TestMaker es una plataforma que se basa para su funcionamiento en la utilización de test de unidad para rehusarlos como pruebas de integración, carga, stress, volumen y regresión. Posee un servicio de vigilancia que se encarga de monitorear el funcionamiento de la aplicación web cada un tiempo que el usuario puede definir. Para la visualización de los resultados cuenta con una gran variedad de gráficas así como un monitoreo del funcionamiento de los recursos como son red, CPU, memoria RAM y otros. Estas pueden ser repartidas en distintos nodos y de esta manera lograr un escenario más real donde poder probar las aplicaciones.

Esta herramienta además puede ser usada en conjuntos con otros lenguajes muy usados en la actualidad como Java, .NET, Python, Groovy, PHP, Ruby para la construcción de los casos de pruebas. Soporta igualmente Servicio Orientado a Objetos (SOA) y AJAX. Los datos para la realización de las pruebas pueden ser obtenidos mediante bases de datos o archivos de texto. (26)

Por todo lo planteado anteriormente se decide utilizar para las pruebas propuestas a nivel de sistema la herramienta PushToTest TestMaker.

Para la realización de inyecciones SQL se propone utilizar el SQLMap, esta es una herramienta libre desarrollada en Python para realizar inyección de código SQL automáticamente. Su objetivo es detectar y aprovechar las vulnerabilidades de inyección SQL en aplicaciones web. Una vez que se detecta una o más inyecciones SQL en el host de destino, el usuario puede elegir entre una variedad de opciones entre ellas, enumerar los usuarios, los hashes de contraseñas, los

privilegios, las bases de datos. Entre sus principales características se pueden enumerar:

- ✓ Soporte completo para MySQL, Oracle, PostgreSQL y Microsoft SQL. Además de estos cuatro sistemas de gestión de bases de datos, sqlMap también puede identificar Microsoft Access, DB2, Informix, Sybase e Interbase.
- ✓ Amplia base de datos de sistema de gestión de huellas dactilares basadas en inband error messages, analizar el banner, las funciones de salida de comparación y características específicas tales como MySQL *comment injection.*
- ✓ Soporte completo para las técnicas de inyecciones SQL: *blind SQL injection* y *inband SQL injection.(28)*

2.6 Recursos Necesarios.

Para llevar a cabo esta estrategia de pruebas se debe contar con una serie de recursos tanto humanos como de hardware, estos deben ser tenidos en cuenta a la hora de crear el grupo de calidad de la facultad.

2.6.1 Recursos Humanos.

Según menciona el profesor Javier Tuya del departamento de informática de la universidad de Oviedo (29), el ingeniero de pruebas debe de tener una serie de habilidades y capacidades específicas que lo ayudaran a un mejor rendimiento en su tarea. Este debe poseer curiosidad e intuición para encontrar situaciones donde se puedan encontrar errores. Además, debe tener la capacidad de actuar bajo presión de tiempo y de conflicto de intereses con el equipo de desarrollo. El ingeniero de pruebas también debe poseer habilidades comunicativas para ser capaz de comunicarle al equipo de desarrollo de forma clara y precisa cuales son los errores encontrados. También se propone la rotación del personal de los proyectos por el equipo de pruebas y el de desarrollo, teniendo en cuenta que un programador que sea capaz de hacer pruebas, será un mejor programador y un

probador con experiencia en el desarrollo de sistemas será un mejor ingeniero de pruebas. La relación equipo de desarrollo - equipo de pruebas en ningún caso debe ser mayor de 4 - 1. Por ello se propone como diseñador de pruebas a un profesor con experiencia en este tipo de trabajo, mientras como probadores se proponen estudiantes de 3er y 4to años que sería muy beneficioso que contaran con experiencia en equipo de desarrollo de software. (29)

2.6.2 Recursos de Hardware.

Los recursos de hardware necesarios para realizar las pruebas varían en función de tamaño del sistema a probar y la cantidad de casos de prueba que este posea, así como del tiempo disponible para realizar todo el proceso que en muchos casos se ve recortado con respecto al propuesto inicialmente durante la planificación del proyecto, debido a que posibles atrasos en otras actividades manteniendo fija la fecha de entrega, conlleva a una reducción del tiempo necesario para hacer las pruebas, esto sucede por la subestimación por parte de algunos líderes de proyectos de este vital proceso.

Por lo anteriormente señalado los recursos de hardware necesarios deben permitir llevar a cabo adecuadamente el proceso de pruebas y en todo momento se deben prever recursos disponibles para asignar al grupo de pruebas con el fin de en un momento determinado poder acelerar el proceso en los casos que así sea requerido.

2.6.3 Recursos de Software.

En el caso del software necesario para realizar la actividad se necesita el sistema operativo que se defina para trabajar, el entorno integrado de desarrollo en el cual se haya desarrollado el sistema a probar y las herramientas definidas para automatizar las pruebas que así lo requieran.

2.7 Conclusiones Parciales.

En este capítulo se describe la Estrategia de Pruebas propuesta evidenciando los niveles, tipos, métodos y técnicas de pruebas que contribuyen a garantizar las

características del software para las telecomunicaciones. Se detallan las actividades comunes que se realizan durante el proceso de pruebas, que fueron identificadas en las metodologías usadas en el Centro, además de los artefactos generados y sus responsables. Se proponen las herramientas a utilizar en cada uno de los niveles estudiados para ayudar en la automatización del proceso y se mencionan los recursos necesarios a tener en cuenta a la hora de llevar a cabo la estrategia.

Capítulo 3: Validación de la Propuesta.

Introducción.

Parte principal de toda investigación es la validación de los resultados obtenidos. Para realizarla existen diferentes modos los cuales llevan a comprobar la garantía de los resultados obtenidos. Entre los métodos existentes se seleccionó el de Expertos debido a que existían las condiciones necesarias para aplicarlo. En este capítulo se analiza de manera estadística las respuestas brindadas por los expertos encuestados con el fin de llegar a conclusiones acerca del criterio generalizado de estos sobre la estrategia propuesta.

3.1 Método de Expertos.

Los métodos de expertos utilizan como fuente de información un grupo de personas a las que se supone poseedoras un conocimiento elevado de la materia que se va a tratar. Estos métodos se emplean cuando se da alguna de las siguientes condiciones (30):

- ✓ No existen datos históricos con los que trabajar. Un caso típico de esta situación es la previsión de implantación de nuevas tecnologías o metodologías.
- ✓ El impacto de los factores externos tiene más influencia en la evolución que el de los internos. Así, la aparición de una legislación favorable y reguladora y el apoyo por parte de algunas empresas a determinadas tecnologías pueden provocar un gran desarrollo de éstas que de otra manera hubiese sido más lento.
- ✓ Las consideraciones éticas o morales dominan sobre las económicas y tecnológicas en un proceso de evolutivo. En este caso, una tecnología puede ver dificultado su desarrollo si éste provoca un alto rechazo en la sociedad (un ejemplo lo tenemos en la tecnología genética, que ve dificultado su avance por los problemas morales que implica la posibilidad de manipulación del genotipo.

Entre las ventajas de los métodos de expertos podemos mencionar:

- ✓ La información disponible está siempre más contrastada que aquella de la que dispone el participante mejor preparado, es decir, que la del experto más versado en el tema. Esta afirmación se basa en la idea de que varias cabezas son mejor que una.
- ✓ El número de factores que es considerado por un grupo es mayor que el que podría ser tenido en cuenta por una sola persona. Cada experto podrá aportar a la discusión general la idea que tiene sobre el tema debatido desde su área de conocimiento.

El método experto ideal debería de extraer los beneficios de la interacción directa e intentara eliminar sus inconvenientes. Ese es el objetivo del método Delphi.

3.2 Método Delphi.

Esta técnica se ajusta favorablemente al tipo de investigación definida y a los objetivos que se persiguen con la misma, si se tienen en cuenta las siguientes consideraciones (30):

- ✓ El método Delphi es una técnica prospectiva de obtención de información cualitativa o subjetiva, pero relativamente precisa en contextos de información imperfecta, fruto de combinar el conocimiento y experiencia de expertos, de una forma que tiende hacia el consenso de opiniones sobre efectos específicos, cuantificando estadísticamente, a su vez, estas opiniones.
- ✓ La idea nuclear es que el conocimiento "grupal" es mejor al de un solo experto en áreas donde la información exacta no se encuentra disponible.
- ✓ Sus rasgos más sobresalientes son el anonimato, la respuesta "grupal" y la tendencia al consenso.

El método Delphi puede utilizarse como instrumento con objetivos diversos como la identificación de problemas, la obtención de información desconocida sobre algún aspecto, la previsión de la evolución de tendencias futuras, entre otros. El

proceso de "creación del cuestionario" y "la selección de los expertos" a participar en el estudio dependerán de los objetivos trazados.

En suma, el método Delphi es un sistema de recogida de información "grupal" de carácter cualitativo, que pretende realizar una recolección de cerebros sobre un conjunto predeterminado de expertos seleccionados a priori, que proporcionen información de manera ordenada y sistemática.

Para el diseño del método Delphi se tomó como referencia la adaptación realizada por Nodal (2003) (31), la cual se modificó de acuerdo con los objetivos de la presente investigación. De modo que el método Delphi se instrumentó siguiendo los siguientes pasos:

1) Consulta a expertos según grupo creado previamente.

2) Desarrollo de una ronda con los expertos seleccionados.

Se decidió realizar una ronda; incluyéndole a las preguntas además la valoración cuantitativa la posibilidad del experto de expresar su criterio sobre la pregunta en cuestión, cuya finalidad fue reducir la dispersión en las informaciones que se pretendían consensuar. Otra faceta fundamental de un estudio basado en el método Delphi es la elaboración del cuestionario que se va a dirigir a los expertos. En el cuestionario que se diseñó para el presente estudio se trataron de recoger las opiniones cualitativas de los participantes bajo criterios de carácter cuantitativo.

3) Aplicación de la escala de Likert: para observar la homogeneidad o dispersión entre las opiniones de los expertos.

Para recoger las observaciones se emplearon preguntas de escala, donde el experto debía asignar una puntuación a cada una de las posibles opciones. Estas son las preguntas cuyo tratamiento cuantitativo y facilidad de respuesta son más sencillos, y donde se puede observar la homogeneidad o dispersión de cada una de ellas. Se aplicó esta técnica agrupando resultados mediante escala de Likert de 4 posiciones. La decisión de emplear una escala de Likert de cuatro posiciones, partió del análisis realizado por San Martín (2005) (32), al recopilar estudios en que se aplicó esta escala, y estas posiciones prevalecieron. Las principales ventajas

de su utilización se basan en la facilidad para administrar y codificar los datos, así como en la posibilidad de someterlos a un tratamiento estadístico.

3.3 Selección de los Expertos.

Se conformó para las encuestas un panel de 8 expertos formados por profesores de la facultad 2 con experiencia en la realización de pruebas de software así como especialistas de calidad de software de la empresa DATYS. Estos fueron seleccionados basados en sus conocimientos sobre calidad y pruebas de software, igualmente por su capacidad de análisis y su pensamiento lógico y por el prestigio ganado en su colectivo de trabajo. Además, para la selección de los expertos se tomaron en cuenta cualidades como la seriedad, la responsabilidad y la honestidad para garantizar de esta forma la confiabilidad de las opiniones brindadas.

3.4 Elaboración del Cuestionario.

Para la elaboración del cuestionario (Anexo 3) se tuvieron en cuenta cada uno de los tópicos abordados durante esta investigación de manera tal que los expertos pudieran emitir su criterio en relación a cada uno de ellos. Los principales componentes de la estrategia de pruebas como los niveles de pruebas, los tipos y técnicas de prueba se evalúan en preguntas independientes con el fin de permitir a los expertos profundizar en sus evaluaciones y opiniones sobre cada uno de estos en particular debido a su importancia para el resultado final de la investigación.

3.5 Coeficiente de competencia de los Expertos.

El método Delphi propone una forma para saber el coeficiente de competencia del conjunto de los expertos seleccionados con el fin de calificar con un valor numérico la confiabilidad brindada por sus criterios. Este coeficiente se determina mediante la suma del coeficiente de conocimientos, el cual brinda a criterio del experto su nivel de conocimientos sobre el tema abordado y el coeficiente de argumentación el cual da la posibilidad de definir cuáles son las fuentes de este conocimiento.

Luego de calculado el coeficiente de competencia para cada experto se determinó que el promedio entre los mismos fue de 0.787 por lo cual se considera como buena la selección de expertos realizada (Anexo 4).

3.6 Procesamiento del Resultado del Cuestionario.

Para el procesamiento del cuestionario se calculó el *valor ponderado* de cada ítem, así como la *media aritmética,* la *desviación típica* y el *coeficiente de variación* para identificar el nivel de dispersión de las opiniones de los expertos en cada pregunta. Vale señalar que los coeficientes de variación calculados no mostraron altos por cientos, todos estuvieron por debajo del 20% con respecto a su media, lo que indica que no hay mucha dispersión entre las opiniones de los expertos. (Anexo 3)

Dentro de la escala de Likert, se decidió establecer los rangos:

- ✓ [1] El experto considera que no cumple las necesidades en esa pregunta.
- ✓ [2] El experto considera que la solución planteada en la pregunta cumple solo en cierto grado las necesidades.
- ✓ [3] El experto considera que la pregunta se cumple adecuadamente.
- ✓ [4] El experto considera que la pregunta se cumple de manera excelente.

A continuación se presentarán los resultados obtenidos de la consulta a expertos: (Anexo 5)

En la pregunta 3 el 88 % de los expertos coincidieron en clasificar como excelentes los niveles de prueba propuestos para formar parte de la estrategia. La media aritmética de la pregunta fue de 3.9 mientras que las respuestas se desviaron solamente en un 8.6 %.

En la pregunta 4 el 50 % de los encuestados consideran adecuada la manera en que la estrategia propuesta tributa a las actividades del flujo de trabajo de pruebas, mientras que el 38 % lo califica como excelente y solo el 13 % lo califica como regular. En esta pregunta la media fue de 3.25 obteniéndose una desviación del 20 % con respecto a la media.

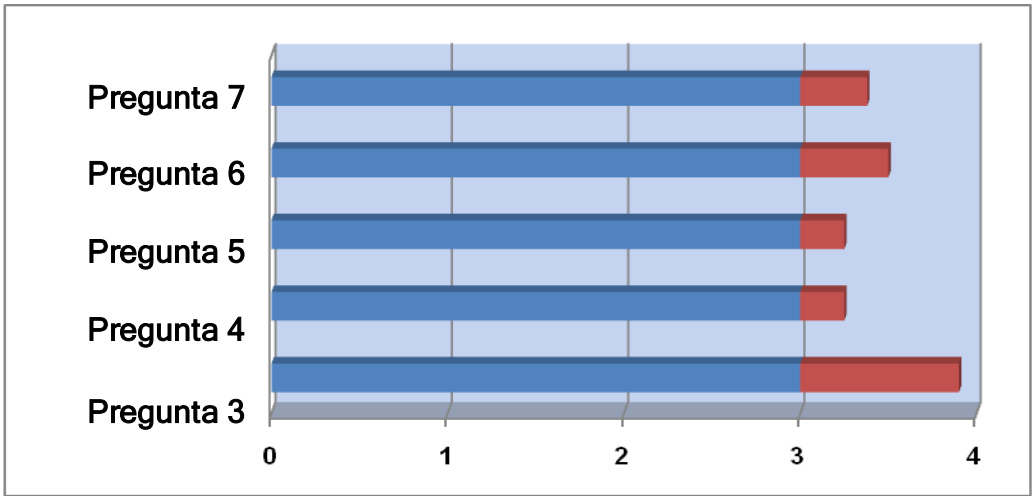
La desviación fue de un 18 % con respecto a la media de 3.25, al ser consultados los expertos sobre los artefactos propuestos en la estrategia y la forma en que estos contribuyen al flujo de trabajo de pruebas. En esta pregunta el 63 % de los mismos los consideró excelente mientras el 37% los catalogó de adecuado.

En la pregunta número 6 sobre cómo consideran que el conjunto de técnicas y herramientas contribuye a la calidad del software del centro un 63 % lo calificaron como excelente mientras que un 25% lo calificó como adecuado; siendo la media de 3.5 y una desviación con respecto a ella del 20%.

En la pregunta referida a la forma en que la estrategia da cumplimiento a las características de calidad identificadas como claves para el software de telecomunicaciones un 75% la califico como excelente. La media en esta pregunta resultó 3.38 con una desviación del 17%.

En la figura 3.1 se muestran una comparación de la medias resultantes en cada una de las preguntas aplicadas.

Figura 3.1: Resultados del Método Delphi.



3.7 Conclusiones Parciales.

Los resultados anteriores demuestran la validez de la estrategia propuesta para ser usada en el Centro de Telemática debido a estar la misma diseñada teniendo en cuenta las características principales de los sistemas para las telecomunicaciones. Como se pudo apreciar la mayoría de los expertos encuestados coinciden en calificar como adecuados o excelentes los principales aspectos que forman parte de la estrategia.

CONCLUSIONES

A modo de conclusión de la investigación:

- ✓ Se entrevistaron a especialistas en el sector de las telecomunicaciones con el fin de identificar las características más relevantes del software.
- ✓ Se realizó el análisis del proceso de pruebas de software en el área de las telecomunicaciones demostrando que en la universidad no existe una estrategia que considere las principales prioridades del software enfocado al sector de las telecomunicaciones.
- ✓ Se estudiaron las metodologías de desarrollo de software utilizadas en el Centro con el objetivo de que la estrategia propuesta pueda ser utilizada por todos los proyectos independientemente de la metodología usada.
- ✓ Se propusieron niveles, tipos, métodos y técnicas de prueba con el fin de verificar y garantizar las características identificadas anteriormente, así como herramientas para llevarlas a cabo.
- ✓ Se validó la utilidad de la estrategia diseñada mediante el criterio de expertos en el tema de las pruebas de software con años de experiencia.

RECOMENDACIONES

- ✓ Aplicar la estrategia propuesta para todos los proyectos del Centro de Telemática de la Facultad 2 con el objetivo de mejorar el proceso de pruebas de software.
- ✓ Capacitar a todo el personal que labora en el Grupo de Calidad con el fin de aumentar la confiabilidad de las pruebas realizadas.
- ✓ Realizar capacitación referente a las Pruebas de Unidad a todos los programadores pertenecientes a los diferentes proyectos productivos del Centro de Telemática.
- ✓ Luego de aplicada la estrategia de deben incluir aspectos de mejoras en los proceso que se identifique puedan ser realizados de manera más eficiente.

REFERENCIAS BIBLIOGRÁFICAS

1. **Hernández, Vismar Santos.** *Estudio sobre la Industria de Software a nivel mundial, caracterización en America Latina y Cuba.* Ciudad de la Habana : UCI, 2008.
2. **Benítez, Juan.** *Las necesidades TIC de las Empresas tienden a las "nubes".*
3. *Consejo de Calidad UCI.* **Calisoft.** 2009.
4. **Calisoft.** *Libro del Diagnóstico 2009 área Facultad 2.* 2009.
5. **Pressman, Roger S.** *Ingenería de Software, un enfoque práctico.* s.l. : Mc Graw Hill, 1998.
6. **Colectivo de Autores**. [En línea] [Citado el: 10 de marzo de 2010.] http://www.angelfire.com/nt2/softwarequality/ISO9126.pdf.
7. **Fernández Medina, Eduardo.** Alarcos. [En línea] [Citado el: 6 de febrero de 2010.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.
8. **Alberto, Molpeceres.** *Comparación entre RUP, XP y FDD.* 2008.
9. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software. .* Madrid : Addison Wesley, 2000.
10. **IBM Corp.** Rational Unified Process.
11. **Thaureaux Martínez, Marlen y Hernández Quintana, Diana Rosa.** Propuesta de procedimiento de pruebas de calidad para la metodología MA-GMPR-UR2. Ciudad de la Habana : s.n., 2009.
12. **Gutiárrez, Sarah, y otros.** *Proceso Basado en Funcionalidades.*
13. **Garbajosa, Juan y Yague, Agustín.** *Las pruebas en metodologías ágiles y convencionales: papeles diferentes.* 2009.
14. **Talby, D, y otros.** *Agile Software Testing in a Large Scale Project.* 2006
15. **Ambler, S.** Agile Adoption Survey . [En línea] 2008. [Citado el: 15 de marzo de 2010.] http://www.ambysoft.com/surveys/agilesFebruary2008.html.

16. The State of Agile Development. [En línea] 2008. [Citado el: 15 de marzo de 2010.] http://www.versionone.com/AgileSurvey.

17. [En línea] [Citado el: 3 de junio de 2010.] http://www.iec.csic.es/criptonomicon/java/quesjava.html.

18. Desarrollo Web. [En línea] [Citado el: 3 de junio de 2010.] www.desarrolloweb.com/articulos/392.php

19. Desarrollo Web. [En línea] [Citado el: 3 de junio de 2010.] www.desarrolloweb.com/articulos/1325.php

20. ***Hernández Aguilar, Violena.*** *Estrategia de Evaluación de Software, Calisoft.* Ciudad de la Habana : s.n., 2009.

21. **Autores, Colectivo de.** *Conferencia 7, Disciplina de Pruebas UCI.* Ciudad de la Habana : s.n., 2009.

22. ***Hernández Aguilar, Violena.*** *Vinculación entre tipos de pruebas y NC ISO 9126-1, Calisoft.* 2009.

23. **Pressman, Roger S.** *Ingenería de Software, un enfoque práctico.* Madrid : s.n., 2001. 5ta Edición.

24. **Pérez Salomón, Omar.** *La reestructuración mundial de las telecomunicaciones.* **Salomón, Omar Pérez.**

25. **OSWAP.** *The Ten Most Critical Web Application Security Risk.* 2010.

26. **Charro Pérez, Karel Alejandro y Licea Castellanos, Andrés Luis.** *Propuesta de Herramientas para Pruebas de Software Automáticas.* Ciudad de la Habana : s.n., 2009

27. **IPCorp Blog**. [En línea] 10 de Septiembre de 2009. [Citado el: 4 de junio de 2010.] http://www.ipcorp.com.ar/blog/2009/09/10/findbugs-un-programa-para-encontrar-bugs-potenciales-en-java/.

28. **Donovan Secret Blog**. [En línea] 22 de noviembre de 2008. [Citado el: 4 de Junio de 2010.] http://dilancreativo.wordpress.com/2008/11/22/sqlmap-herramienta-automatica-de-inyeccion-sql/.

29. **Tuya, Javier.** *Recursos Humanos para las pruebas de software.* s.l. : Universidad de Oviedo, 2007.

30. **Monfort, V.** *El método Delphi en la Investigación e Información en el Turismo.* s.l. : Universidad de Valencia, 1999.

31. **Nodal, L.** *Mercado de Eventos de las Asociaciones Médicas Internacionales radicadas en Estados Unidos de América: Evaluación de su Actividad.* Ciudad de la Habana : s.n.

32. **Martín, H. San.** *Estudio de la imagen del destino turístico y el proceso global de satisfacción: adopción de un enfoque integrador.* Santander : Universidad de Cantabria.

BIBLIOGRAFÍA

1. **Acebal, Cesar F. y Cueva Lovelle, Juan M.** *Extreme Programming, un nuevo método de desarrollo de software.*
2. **AT&T.** [En línea] [Citado el: 2010 de Marzo de 25.] http://www.att.com.
3. **Charro Pérez, Karel Alejandro y Licea Castellanos, Andrés Luis.** *Propuesta de Herramientas para pruebas de software automáticas.* Ciudad de la Habana : s.n., 2009.
4. **Colectivo de Autores.** Lista de Herramientas para el testeo de aplicaciones web. [En línea] 20 de Octubre de 2006. [Citado el: 4 de Mayo de 2010.] http://sentidoweb.com/2006/10/20/lista-de-herramientas-para-testeo-de-aplicaciones-web.php..
5. **Colectivo de Autores.** Presentación de la técnica DELPHI. [En línea] [Citado el: 6 de mayo de 2010.] http://www.geocities.com/Pentagon/Quarters/7578/pros01.html.
6. **Corp, IBM.** *Rational Unified Process.* 2006.
7. **Cortes, Oscar H. Guzmán.** *Aplicación Práctica del diseño de pruebas de software a nivel de programación. .* 2004.
8. **Grimán, Anna C., Pérez, María y Mendoza, Luis E.** *Estrategia de Pruebas para Software OO que garantiza requerimientos no funcionales.* Caracas : Laboratorio de Investigación de Sistemas de Información. Universidad Simón Bolivar .
9. **Grupo Alarcos Universidad de Castilla.** Alarcos. [En línea] [Citado el: 2 de Marzo de 2010.] http://alarcos.inf-cr-uclm.es/doc/ISOFTWAREI/Tema9.pdf.
10. **IEEE.** *Standard Glossary of Software Engineering Terminology.* 1990.
11. **ISO.** Normas de gestión de calidad y garantía de la calidad. . [En línea] [Citado el: 11 de diciembre de 2010.] http:campus.fortunecity.com/defiant/114/iso9000.
12. **Jiménez, Darwin y Aguirre, Carlos Eduardo.** *Modelo de Pruebas de Software.*

13. **Letelier, Patricio.** Departamento de Sistemas Informáticos y Computación. [En línea] Universidad Politécnica de Valencia. [Citado el: 19 de febrero de 2010.] www.dsic.upv.es.

14. **Medina, Eduardo Fernández.** Alarcos. [En línea] [Citado el: 6 de Marzo de 2010.] http://alarcos.inf-cr.uclm.es/doc/ISOFWAREI/Tema04.pdf.

15. **Motorola.** [En línea] [Citado el: 2010 de Marzo de 25.] http://www.motorola.com.

16. **OSWAP.** *The Ten most critical web application Security Risks.* 2010.

17. **Pressman, Roger S.** *Ingenería de Software, un enfoque práctico 5ta edición.* Madrid : s.n., 2001.

18. **Pressman, Roger S.** *Ingenería de Software, un enfoque práctico.* s.l. : Mc Graw-Hill, 1998.

19. **Santos Lebeque, Adriana y Hernández Valladares, Danaysi.** *Estrategia de Juegos para Realidad Virtual.* Ciudad de la Habana : s.n., 2008.

20. **Sony Ericcson.** [En línea] [Citado el: 2010 de Marzo de 25.] http://www.sonyericcson.com.

21. **Soriano, A.** Tipos de Pruebas. [En línea] [Citado el: 18 de abril de 2010.] http://carolina.terna.net/ingsw3/datos/Tipos_Prueba.pdf.

22. **TELTRONIC.** [En línea] [Citado el: 2010 de Marzo de 25.] http://www.teltronic.es.

23. **Telefónica.** [En línea] [Citado el: 2010 de Marzo de 25.] http://www.telefonica.es.

24. **Thaureaux Martínez, Marlen y Hernández Quintana, Diana Rosa.** *Propuesta de procedimiento de pruebas de calidad para la metodología MAGMPR-UR2.* Ciudad de la Habana : s.n., 2008.

25. **UCI.** Dirección de Calidad de Software. [En línea] [Citado el: 25 de enero de 2010.] http://calisoft.prod.uci.cu.

26. **Valverde, Jorge C.** *Acerca de las estrategias de pruebas de software.* Santo Domingo : Universidad Nacional de Trujillo, 2006.

27. **Vegas, Sira, Juristo, Natalia y Moreno, Ana M.** Técnicas de Evaluación de Software . [En línea] 17 de octubre de 2005. [Citado el: 10 de abril de 2010.] http://is.ls.fi.upm.es/docencia/erdsi/Documentacion_Evaluacion_7.pdf

ANEXOS

Anexo 1 Entrevista realizada a especialistas en el desarrollo de software para las telecomunicaciones.

GUÍA PARA ENTREVISTA A ESPECIALISTAS EN EL DESARROLLO DE SOFTWARE PARA LAS TELECOMUNICACIONES SOBRE LAS CARACTERÍSTICAS SIGNIFICATIVAS DE ESTE TIPO DE SISTEMA.

Febrero, 2010. "Año del 51 aniversario del Triunfo de la Revolución Cubana"

Menciones de las características de calidad presentes en la NC ISO/IEC 9126-1:2005 y sus sub-características, cuáles cree usted sean más importante para el software desarrollado para las telecomunicaciones. Argumente.

Funcionalidad: se define como un conjunto de atributos que atañen a la existencia de un conjunto de funciones y sus propiedades específicas. Estas funciones son las que satisfacen las necesidades implícitas y establecidas. Esta característica del software puede ser desglosada en varias sub-características:

- ✓ Adecuación: no es más que la capacidad del software de proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos del usuario.
- ✓ Exactitud: puede definirse como la capacidad del software para proporcionar resultados correctos o que necesitan un determinado grado de precisión.
- ✓ Interoperabilidad: es la capacidad del software de interaccionar con uno o más sistemas especificados.
- ✓ Seguridad: definida como la capacidad del software de proteger la información y los datos.

Confiabilidad: se define como el conjunto de atributos que atañen a la capacidad del software para mantener su nivel de prestación bajo condiciones establecidas durante un tiempo establecido. Se descompone en las siguientes características:

- ✓ Madurez: es la capacidad del software para evitar fallos como resultados de defectos del software.
- ✓ Tolerancia a fallos: capacidad del software para mantener un nivel especificado de rendimiento en casos de fallos del software.
- ✓ Capacidad de recuperación: capacidad para restablecer el nivel de rendimiento y de recuperación de datos afectados directamente en el caso de un fallo.

Facilidad de uso: se entiende como la capacidad del producto software de ser entendido, aprendido, usado y atraer al usuario, cuando es utilizado bajo ciertas condiciones específicas. Se descompone en:

- ✓ Fácil comprensión: se entiende como la capacidad del software que permite al usuario si el producto es aceptable y cómo puede ser usado para tareas particulares y determinadas condiciones de uso.
- ✓ Fácil aprendizaje: capacidad del producto software que permite al usuario aprender la aplicación software.
- ✓ Operatividad: capacidad del producto software que permite al usuario controlar y usar la aplicación software.
- ✓ Software atractivo: capacidad del producto software de ser atractivo al usuario.

Eficiencia: está dada por la capacidad del producto software para proporcionar un rendimiento apropiado relacionado con el total de recursos utilizados bajo condiciones establecidas. Se subdivide en las siguientes características.

- ✓ Comportamiento frente al tiempo: capacidad del producto software para proporcionar una respuesta y un tiempo de procesamiento apropiados al desarrollar sus funciones bajo condiciones establecidas.
- ✓ Uso de recursos: capacidad del producto software para utilizar un apropiado número de recursos y tiempo de ejecución cuando el software desarrolla sus funciones bajo condiciones establecidas.
- ✓ Adherencia a normas: capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes en leyes o prescripciones similares.

Mantenibilidad defina como la capacidad del producto software para ser modificado. Se descompone en las siguientes características.

- ✓ Facilidad de análisis: capacidad del producto software para diagnosticar deficiencias o causas de fallos en el software.
- ✓ Capacidad para cambios: capacidad del producto software que permite la ejecución de una modificación específica en ella misma.
- ✓ Estabilidad: capacidad del producto de software para evitar defectos no esperados debidos a modificaciones en el mismo.
- ✓ Facilidades para pruebas, capacidad del producto software que permite al software que ha sido modificado, ser evaluado.

Portabilidad se entiende como la capacidad del producto software para ser transferido de un entorno a otro. El entorno se interpreta tanto a nivel software y hardware, como aquel entorno relacionado con la organización. Se divide en:

- ✓ Adaptabilidad: capacidad del producto software para ser adaptado a diferentes entornos especificados sin aplicar acciones alejadas de aquellas que el propio software proporcione.
- ✓ Facilidad de instalación: capacidad del producto software para ser instalado en un entorno específico.

- ✓ Coexistencia: capacidad del producto software de coexistir con otros programas independientes en un entorno común y compartiendo recursos también comunes.
- ✓ Facilidad de reemplazo, capacidad del producto software de ser utilizado en lugar de otro producto software específico para el mismo propósito que éste y en un entorno similar.

Anexo 2. Plan de Pruebas utilizado por la VPTI de ETECSA.

<Nombre del proyecto>
Plan de Pruebas

Versión <0.1>

[Nota: La siguiente plantilla se ha desarrollado para su uso en los proyectos de desarrollo de software de la CASA SIGTA. El texto que se encuentra entre corchetes y presentado en estilo itálico azul se ha incluido para proporcionar una guía para el autor y se debería borrar antes de la entrega del documento.]

[Hay que sustituir el texto resaltado con marcador amarillo por su equivalente en el proyecto de desarrollo y eliminar el resaltado]

[La versión del documento se actualizará según la iteración y la fase del proyecto]

Historial de Revisiones

Fecha	Versión	Descripción	Autor
<dd/mm/aa>	<0.1>	<Versión preliminar como propuesta de desarrollo. >	< Autor>

Tabla de Contenidos

PLAN DE PRUEBAS

Introducción

La elaboración de un plan de pruebas adecuado es la piedra angular y principal factor crítico de éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel.

Todos los proyectos, ya sean nuevos desarrollos o mantenimientos, aunque tienen un marco común para la realización de las pruebas, llevan un proceso diferente que debe ser planificado previo a su realización.

Propósito

El Plan de Pruebas tiene como objetivo establecer las pruebas y revisiones que se van a ejecutar en cada fase, así como los cronogramas y responsables de ejecución.

Alcance

[Breve descripción del alcance del plan de pruebas, las fases de pruebas excluidas, a que proyecto se asocia y cualquier otra cosa que sea afectado o influenciado por este documento].

Definiciones, Siglas, y Abreviaturas

[Esta subsección ofrece la definición de todos los términos, siglas y abreviaciones que se necesitan para interpretarlo adecuadamente. Esta información puede ser proporcionada por referencia al Glosario del proyecto]

Referencias

[Esta subsección ofrece una lista numerada de todos los documentos referidos ordenados alfabéticamente por los apellidos de los autores. Identifica cada documento por un consecutivo, nombre de autores, título, fecha y organizaciones y/o editoriales. Esta información puede ser proporcionada por referencia a un apéndice o a otro documento.]

Este documento hace referencia a:

Apellido, Nombre. Documento <Nombre del Documento>. <mes, año>. <organización o editorial>.

Apellido, Nombre. Documento <Nombre del Documento>. <mes, año>. <organización o editorial>.

Documentación a revisar.

El siguiente listado establece los documentos entregables que deben ser revisados por el Grupo de Calidad de la Gerencia en cada una de las etapas del proyecto.

Documento Entregable	Etapa del proyecto	Fecha de entrega	Responsable de revisión	Observaciones.

Fases de Pruebas.

[Se listaran las fases de pruebas que se van a ejecutar en el proyecto con el nombre del responsable de su ejecución así como el estimado de duración de tiempo para cada una.]

Fase de Prueba	Responsable de ejecución	Fecha de inicio	Fecha de Culminación	Observaciones.

Fase de prueba <Nombre de la fase 1>.

Se listaran las pruebas que se ejecutaran dentro de cada fase definida. Se establece además, el alcance de las pruebas que se van a ejecutar en el proyecto con el nombre del responsable de su ejecución así como la duración de tiempo para cada una].

Tipos de Prueba	*Alcance*	*Responsable de ejecución*	*Duración*	*Observaciones*

Fase de prueba <Nombre de la fase 2>.

Se listaran las pruebas que se ejecutaran dentro de cada fase definida. Se establece además, el alcance de las pruebas que se van a ejecutar en el proyecto con el nombre del responsable de su ejecución así como la duración de tiempo para cada una].

Tipos de Prueba	*Alcance*	*Responsable de ejecución*	*Duración*	*Observaciones*

Fase de prueba <Nombre de la fase 3>.

Se listaran las pruebas que se ejecutaran dentro de cada fase definida. Se establece además, el alcance de las pruebas que se van a ejecutar en el proyecto con el nombre del responsable de su ejecución así como la duración de tiempo para cada una].

Tipos de Prueba	*Alcance*	*Responsable de ejecución*	*Duración*	*Observaciones*

Fase de prueba <Nombre de la fase 4>.

Se listaran las pruebas que se ejecutaran dentro de cada fase definida. Se establece además, el alcance de las pruebas que se van a ejecutar en el proyecto con el nombre del responsable de su ejecución así como la duración de tiempo para cada una].

Tipos de Prueba	*Alcance*	*Responsable de ejecución*	*Duración*	*Observaciones*

Fase de prueba <Nombre de la fase 5>.

Se listaran las pruebas que se ejecutaran dentro de cada fase definida. Se establece además, el alcance de las pruebas que se van a ejecutar en el proyecto con el nombre del responsable de su ejecución así como la duración de tiempo para cada una].

Tipos de Prueba	*Alcance*	*Responsable de ejecución*	*Duración*	*Observaciones*

Lista de chequeo de ejecución del plan de pruebas.

El cumplimiento del cronograma planificado se registra en esta lista de chequeo (incluye las revisiones a la documentación y las pruebas a las aplicaciones).

Entregable bajo prueba	**Tipo de prueba**	**Fase**	**Aprobación**	**Fecha de cumplimiento**	**Observaciones**
Se refiere al documento o módulo revisado	*Tipo de prueba dentro de cada fase*	*Fase de prueba*			

ANEXO 3: Cuestionario de evaluación de la Propuesta de estrategia de pruebas de software para el Centro de Telemática (SITEL)

Usted ha sido seleccionado para participar en una encuesta con el fin de validar la Propuesta de estrategia de pruebas de software para el Centro de Telemática. Usted ha sido seleccionado debido a sus conocimientos en cuanto al proceso de pruebas de software. Esta validación se lleva a cabo por el método Delphy, el cual adicionalmente a las preguntas relacionadas con el tema a estudio propone una serie de preguntas relacionadas con su conocimiento sobre el tema y la forma que ha sido adquirido.

Datos

Nombre y apellidos:

Proyecto:

Labor que realiza:

Años de experiencia:

1 Marque el grado de conocimiento que usted considera tener en cuanto al proceso de pruebas de software.

0	1	2	3	4	5	6	7	8	9	10

2 Marque cuáles han sido las fuentes principales de las cuales ha adquirido su conocimiento sobre el tema.

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted			
Su experiencia obtenida			
Trabajos de autores nacionales			
Trabajos de autores extranjeros			
Su propio conocimiento del estado del problema en el extranjero			
Su intuición			

3 Cómo considera usted la selección de los niveles de prueba usados en la estrategia teniendo en cuenta las necesidades del software para las telecomunicaciones.

Justifique.

___ Excelente ___ Regular

___Adecuado ___ No cumple las necesidades

_____

_____

_____

_____

_____

4 Clasifique en qué grado considera usted que la estrategia propuesta tributa a las actividades del Flujo de trabajo de Prueba, independientemente de la metodología de desarrollo usada.
Justifique.

___ Excelente ___ Regular
___Adecuado ___ No cumple las necesidades

_____
_____
_____
_____
_____

5 Cómo considera usted que los artefactos propuestos en la estrategia complementan las actividades del Flujo de trabajo de Pruebas.
Justifique.

___ Excelente ___ Regular
___Adecuado ___ No cumple las necesidades

_____
_____
_____
_____
_____

6 Cómo considera que el conjunto de técnicas y herramientas descritas en la estrategia de pruebas garantizan la calidad del software desarrollado en el Centro SITEL.
Justifique.

___ Excelente ___ Regular
___Adecuado ___ No cumple las necesidades

_____
_____
_____
_____
_____

7 ¿En qué grado califica usted que la estrategia da cumplimiento a las características de calidad identificadas para el software desarrollado en el Centro? Justifique.

___ Excelente ___ Regular
___Adecuado ___ No cumple las necesidades

_____
_____

_____

_____

_____

Muchas gracias por su colaboración, ha sido de gran ayuda para esta investigación.

ANEXO 4: Cálculo de los Coeficientes de Expertos.

Valor de los Criterios:

FUENTES DE ARGUMENTACION	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted	0.3	0,2	0,1
Su experiencia obtenida	0.5	0,4	0,2
Trabajos de autores nacionales	0.05	0,04	0,03
Trabajos de autores extranjeros	0.05	0,04	0,03
Su propio conocimiento del estado del problema en el extranjero	0.05	0,04	0,03
Su intuición	0.05	0,04	0,03
Totales	1.0	0,76	0,42

Experto 1

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted		x	
Su experiencia obtenida		x	
Trabajos de autores nacionales	x		
Trabajos de autores extranjeros	x		
Su propio conocimiento del estado del problema en el extranjero			x
Su intuición		x	

Ka= 0,2+0,4+0,05+0,05+0,04+0,03=0,77 kc=0,7

K=1/2(ka+kc)=0,735

Experto 2

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted		x	
Su experiencia obtenida		x	
Trabajos de autores nacionales			x
Trabajos de autores extranjeros	x		
Su propio conocimiento del estado del problema en el extranjero			x
Su intuición		x	

Ka= 0,2+0,4+0,03+0,05+0,03+0,04=0,75 kc=0,7

K=1/2(ka+kc)=0,725

Experto 3

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus

	criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted	X		
Su experiencia obtenida		x	
Trabajos de autores nacionales		x	
Trabajos de autores extranjeros		x	
Su propio conocimiento del estado del problema en el extranjero			x
Su intuición		x	

Ka= 0,3+0,4+3(0,04)+0,03=0,85

kc=0,8

K=1/2(ka+kc)=0,825

Experto 4

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted	X		
Su experiencia obtenida	x		
Trabajos de autores nacionales			x
Trabajos de autores extranjeros		x	
Su propio conocimiento del estado del problema en el extranjero			x
Su intuición		x	

Ka= 0,3+0,5+0,03+0,04+0,03+0,04=0,94 kc=0,8

K=1/2(ka+kc)=0,87

Experto 5

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted			x
Su experiencia obtenida		x	
Trabajos de autores nacionales		x	
Trabajos de autores extranjeros		x	
Su propio conocimiento del estado del problema en el extranjero			x
Su intuición		x	

Ka=0,1+0,4+0,04+0,04+0,03+0,04=0,65 kc=0,7

K=1/2(ka+kc)=0,675

Experto 6

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.

	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted		x	
Su experiencia obtenida	x		
Trabajos de autores nacionales		x	
Trabajos de autores extranjeros			x
Su propio conocimiento del estado del problema en el extranjero		x	
Su intuición		x	

Ka=0,2+0,5+0,04+0,03+0,04+0,04=0,85 kc=0,8

K=1/2(ka+kc)=0,825

Experto 7

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A	M	B

	(alto)	(medio)	(bajo)
Análisis teóricos realizados por usted	x		
Su experiencia obtenida		x	
Trabajos de autores nacionales		x	
Trabajos de autores extranjeros			x
Su propio conocimiento del estado del problema en el extranjero			x
Su intuición		x	

Ka=0,3+0,4+0,04+0,03+0,03+0,04=0,84 kc=0,8

K=1/2(ka+kc)=0,82

Experto 8

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)

Análisis teóricos realizados por usted		x	
Su experiencia obtenida		x	
Trabajos de autores nacionales			x
Trabajos de autores extranjeros	x		
Su propio conocimiento del estado del problema en el extranjero			x
Su intuición		x	

Ka=0,2+0,4+0,04+0,03+0,03+0,04=0,74 kc=0,9

K=1/2(ka+kc)=0,82

Promedio de K=0,787

ANEXO 5: Procesamiento de los Cuestionarios del Método Delphi.

Preg./Exp	1	2	3	4	5	6	7	8	Suma E	Media	Varianza	Desviación Típica	Coeficiente de Variación
Pregunta 3	4	4	4	4	4	3	4	4	31	3,9	0,11	0,331662479	0,085590317
Pregunta 4	3	2	4	4	3	3	3	4	26	3,25	0,4375	0,661437828	0,203519332
Pregunta 5	4	4	4	3	3	4	4	3	29	3,25	0,38	0,612372436	0,188422288
Pregunta 6	3	3	4	4	2	4	4	4	28	3,50	0,5	0,707106781	0,202030509
Pregunta 7	3	4	4	4	3	4	4	4	30	3,38	0,32	0,569561235	0,168758884

Pregunta 3

Desv

xi	ni	frec A A	frec Rel	Frec rel A	xi * ni	xi-xm	Des()2	ni*des()2
0	0	0	0	0	0	-3,9	15,21	0
1	0	0	0	0	0	-2,9	8,41	0
2	0	0	0	0	0	-1,9	3,61	0
3	1	1	0,13	0,13	3	-0,9	0,81	0,81
4	7	8	0,88	1	28	0,1	0,01	0,07
Total	8				31			0,88

Pregunta 4

Desv

xi	ni	frec A A	frec Rel	Frec rel A	xi * ni	xi-xm	Des()2	ni*des()2
0	0	0	0	0	0	-3,25	10,5625	0
1	0	0	0	0	0	-2,25	5,0625	0
2	1	1	0,13	0,13	2	-1,25	1,5625	1,5625
3	4	5	0,50	0,63	12	-0,25	0,0625	0,25
4	3	8	0,38	1	12	0,75	0,5625	1,6875
Total	8				26			3,5

Pregunta 5

Desv

xi	ni	frec A	frec	Frec rel	xi * ni	xi-xm	Des()2	ni*des()2

		A	Rel	A				
0	0	0	0	0	0	-3,25	10,5625	0
1	0	0	0	0	0	-2,25	5,0625	0
2	0	0	0	0	0	-1,25	1,5625	0
3	3	3	0,38	0,38	9	-0,25	0,0625	0,1875
4	5	8	0,63	1	20	0,75	0,5625	2,8125
Total	8				29			3

Pregunta 6

Desv

xi	ni	frec A A	frec Rel	Frec rel A	xi * ni	xi-xm	Des()2	ni*des()2
0	0	0	0	0	0	-3,5	12,25	0
1	0	0	0	0	0	-2,5	6,25	0
2	1	1	0,13	0,13	2	-1,5	2,25	2,25
3	2	3	0,25	0,38	6	-0,5	0,25	0,5
4	5	8	0,63	1	20	0,5	0,25	1,25
Total	8				28			4

Pregunta 7

Desv

xi	ni	frec A	frec	Frec rel	xi * ni	xi-xm	Des()2	ni*des()2

		A	Rel	A				
0	0	0	0	0	0	-3,38	11,4244	0
1	0	0	0	0	0	-2,38	5,6644	0
2	0	0	0	0	0	-1,38	1,9044	0
3	2	2	0,25	0,25	6	-0,38	0,1444	0,2888
4	6	8	0,75	1	24	0,62	0,3844	2,3064
Total	8				30			2,5952