Universidad de las Ciencias Informáticas

Facultad 8

Título: Propuesta de un Proceso de Prueba para un proyecto de Software de Gestión.

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

Autoras: Telma Rodríguez Alfonso
Adisleydis Olano Montero

Tutor: MSc. Yamilis Fernández Pérez

"Ciudad de la Habana. Julio, 2007"

Declaración de Autoría

Declaramos ser autoras de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

"Telma Rodríguez Alfonso" “Adisleydis Olano Montero”

_____

“Yamilis Fernández Pérez”

Datos de Contacto

Tutora: Yamilis Fernández Pérez

Breve Currículo

- Profesora de Ingeniería de Software.
- Graduada de Ingeniera en Sistema Automatizado de Dirección, en 1992 en el ISPJAE.
- Profesora asistente desde 1995.
- MSc. en Informática Aplicada en 1995.
- Imparte docencia en universidades desde 1992.
- Ha desarrollado trabajos con Universidades extranjeras en Brasil, Bolivia, Canadá.
- Es jefa del departamento docente central de Ingeniería y Gestión de Software de la UCI desde su fundación.
- Tiene certificado el curso de Introducción al CMMI que imparte el SEIS.

Ubicación: UCI, Cuba
e-mail: yamilisf@uci.cu

“La lucha por la calidad del producto, es una lucha revolucionaria y de vanguardia”

Che

Agradecimientos

A nuestros padres, por ser una excelente guía en el camino de la vida, y por tener fe en nosotras, no los defraudaremos.

A nuestra tutora Yamilis Fernández, por tener tanta paciencia y por todo el empeño que puso en ayudarnos.

A nuestros profesores, por tratar de hacer de nosotras unas excelentes profesionales.

A Alcides Cabrera, Raciel Mapolón, a Mijail A. Saralain, a Juan Carlos Granja, y todos aquellos que de un modo u otro nos han apoyado para hacer realidad este sueño.

A nuestras amistades, por apoyarnos en todo momento y hacernos más felices cada días.

A nuestros familiares, gracias por el apoyo en esta etapa tan importante y difícil.

A todas aquellas personas que de un modo u otro han hecho posible que este proyecto sea una linda verdad.

Dedicatoria

De Adis:

A mi mamá Silvia Montero, por haberme dado la vida y ser la luz de mis días.

A mi papá Modesto Olano por ser un ejemplo de dedicación y por enseñarme a no claudicar.

A mis hermanas y mis sobrinos, por su apoyo y por quererme tanto.

A mis amigas de la Universidad y de toda la vida Karen, Aine, Yele, Anet, Alieny, Dariena, quienes en estos años me comprendieron y apoyaron en todo momento, nunca las voy a olvidar.

A ti, por haberme alegrado la vida.

A mis compañeras de cuarto Blanca, Yennia, Yanet, Isis, Yini, Lilian, Yilennis, Misde, Daily, Greisy y Celia, por todo este tiempo compartido, las quiero mucho.

A Zuci, Yetel, Lenia, Greisis, mis amigas de toda la vida.

A Telma, gracias por la paciencia y suerte en la vida.

De Telma:

A mi mamá Maura Alfonso, por quererme y apoyarme cada día.

A mis hermanos Alejandro y Gretell, los quiero un montón.

A mis tías, gracias por el apoyo brindado en todos estos años.

A todos mis amigos, por llenar un pedacito de mi corazón.

A Adis, por tener siempre la razón.

A Abd por siempre enseñarme todo y siempre estar a mi lado.

A mis vecinos porque siempre se preocuparon por mí.

Resumen

Los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI), desarrollan productos de software para la exportación y para satisfacer necesidades nacionales, buscando con ello alcanzar un puesto cimero en el mercado de software a nivel mundial. Por ello se hace necesario asegurar que los productos que se obtengan, estén libres de fallos y cumplan a cabalidad las expectativas de los clientes. En el presente trabajo se hace la propuesta de un proceso de pruebas para proyectos de software de gestión, en el marco de la universidad. La propuesta incluye dos cursos de capacitación al personal involucrado en dicho proceso, en aras de desarrollar en ellos las habilidades requeridas para implementarlo. El proceso toma como referencia el proceso de pruebas definido por Rational Unified Process (RUP) y el modelo V. La propuesta fue validada a través del criterio de un panel expertos, conformado por especialistas de la UCI y de la Empresa Softel.

Palabras Claves

Aseguramiento de la Calidad, Pruebas de Software, Proceso de Pruebas para Software de Gestión, Control de la Calidad, Modelos de Pruebas.

Abstract

The productive projects in the University of Informatics Sciences (UIC) develop software products for the exportation and to satisfy national necessities, in order to reach an important position in the software market al world level. With that purpose, it becomes necessary to assure that the products that are obtained will be free of faults and will completely meet the client's expectations. The proposal of a tests process for projects of management software, in the mark of the university, is made in this work. The proposal includes two training courses for the personnel involved in this process, so they can develop the abilities required to implement it. The process takes into account the test process defined by Rational Unified Process (RUP) and the V model. The proposal was validated through the approach of an expert's panel, conformed by specialists of the UIC and of the Softel Company.

Índice

Índice de Figuras

Índice de Tablas

Introducción

En esta era que vive el mundo, donde la mayoría de las actividades humanas pueden ser facilitadas a través de sistemas tecnológicos, la informática, y con ella la producción de software, alcanza cada vez mayor aceptación y demanda a nivel global. El desarrollo alcanza niveles indescriptibles, y crece enormemente la demanda de software mejor y más complejo en menos tiempo y costo.

Cuba no ha sido una excepción en este sentido, y se le concede cada vez mayor importancia a la rama de la informática, no solo por las implicaciones sociales que esta indiscutiblemente tiene, sino también en el sentido económico, en vistas de que es un objetivo primordial del estado y la sociedad cubana insertarse en el mercado informático mundial, y con esto avanzar en el reforzamiento de la economía nacional.

La Universidad de las Ciencias Informáticas (UCI) pretende ser el futuro líder de la producción y exportación de software del país, para ello, en esta institución se realizan anualmente diversos software para numerosas empresas, tanto nacionales como extranjeras. Pero aún necesitan madurar sus procesos de desarrollo de software, y un ejemplo de esto es que se han detectado problemas con los procesos de prueba a software de gestión, los cuales son notables ya casi en el despliegue, y en ocasiones dan al traste con defectos que son solo visibles una vez que el producto está en manos de los clientes, lo cual le resta fiabilidad a la universidad, y por tanto a los productos que esta comercializa. Esto se vuelve un problema, debido a que con tanta competencia, los clientes cada vez exigen mayor precisión, rapidez y flexibilidad en los sistemas informáticos que compran, y por supuesto, acuden entonces a las empresas más confiables para solicitar sus productos.

Es una realidad que en todo proceso de desarrollo de software, las personas que trabajan en el mismo cometen errores, es por ello que se hace necesario seguir un proceso de pruebas que permita detectar y corregir los errores surgidos durante el desarrollo del software, y de esta forma ganar la aceptación del producto por parte de los clientes.

El objetivo general de este trabajo es definir un proceso de pruebas aplicable a los software de gestión que se producen en la UCI, de manera que se logre reducir al

mínimo el número de errores de los mismos, y por tanto, garantizar en cierta medida que tengan un nivel adecuado de calidad con un costo y tiempo mínimo.

Para lograr este objetivo se ha de estudiar el proceso de desarrollo de software según las características de los proyectos productivos de la UCI, y más específicamente, la ingeniería de pruebas en este proceso.

Con el fin de lograr alcanzar el objetivo general, se tuvieron en cuenta los siguientes objetivos específicos:

- Investigar los procesos de pruebas y sus características generales.
- Caracterizar la producción de software en la universidad.
- Diseñar un proceso de pruebas aplicable a los software de gestión que se producen en la UCI.
- Diseñar dos cursos optativos para capacitar personal a la hora de la puesta en práctica del proceso definido.

Se realizaron las siguientes tareas para dar cumplimiento a los objetivos anteriores:

- Estudiar los tipos de pruebas a software que se realizan a nivel mundial, y sus características.
- Entrevistar a directivos del departamento de calidad central, así como a líderes de proyecto de software de gestión para conocer las características de la producción en la universidad y el estado actual de las pruebas a software de gestión.
- Determinar y profundizar en los elementos de las pruebas a software que se realizan a nivel global y que se corresponden con las características de producción de software propias de la universidad.
- Definir un proceso de pruebas para proyectos de software de gestión, de manera que se oriente como llevar a cabo las mismas y obtener los artefactos correspondientes.
- Definir dos cursos de capacitación para el personal involucrado en el proceso de prueba.

Dentro de las metodologías informáticas se utilizó RUP, que es una metodología muy potente para el desarrollo de software. Se escogió la misma puesto que consta de varios que aspectos que la hacen ser una metodología bastante completa, que tiene como características principales que guía un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Pretende implementar las mejores prácticas en ingeniería de software, con el objetivo de asegurar la producción de software de calidad, dentro de plazos y presupuestos predecibles.

Esta metodología describe, además, como reutilizar los componentes existentes o implementar nuevos componentes con tareas bien definidas, a través del cual se puede obtener un sistema fácil de mantener, al que se le puede ir incrementando las posibilidades de reutilización. Asegurando la producción de software con el número mínimo de errores se logra un software de alta calidad en un corto tiempo, y por lo tanto, con un mínimo de costo.

Para llevar a cabo la investigación se definió la utilización de métodos teóricos, los cuales posibilitaron el desarrollo del contenido esencial de la investigación, puesto que los mismos permitieron conocer el estado del arte de la ingeniería de prueba, su evolución y relación con otros elementos del proceso de desarrollo de software. Se utilizó en este trabajo el método del análisis histórico lógico porque permitió conocer la evolución y desarrollo de las pruebas a software a lo largo de toda su historia. El análisis de toda lo documentación encontrada, se realizó a través de la utilización del método analítico-sintético, pudiendo con el mismo sintetizar la información para obtener los aspectos más importantes contenidos en todos los documentos que se analizaron. La modelación permitió la definición de un modelo o estrategia para investigar la realidad de la ingeniería de pruebas en el mundo actualmente, y descubrir nuevas cualidades y relaciones de la misma.

El uso de algunos de los métodos empíricos ofreció la posibilidad de poder efectuar un análisis anticipado de la información, también se pudo verificar y comprobar varias concepciones teóricas agilizando la obtención de las principales características del objeto de estudio definido en la investigación. En el marco de estos métodos el más apropiado para la investigación que se realizó fue la encuesta, pues por mediación de este método se pudo obtener información verídica de la situación de la ingeniería de pruebas en proyectos de software de gestión en la universidad, así como de las necesidades reales de la misma.

El trabajo se estructuró en tres capítulos. En el primer capítulo, “Fundamento de las pruebas” se hace un recuento del estado del arte de la ingeniería de pruebas a nivel mundial. El segundo, “Diseño del proceso de pruebas” contiene una caracterización de la producción de software en la universidad, con respecto al tema de las pruebas, y propone un proceso de pruebas para aplicar en los proyectos de producción de software de gestión de la UCI y además contiene los dos cursos de capacitación que se le dará al personal involucrado en las pruebas a software. Finalmente el capítulo 3, “Evaluación del proceso de pruebas” muestra la validación del proceso propuesto, a través del Criterio de un Panel de Expertos.

Capítulo 1 Fundamento de las Pruebas

Introducción

Una de las etapas más importantes en el proceso de desarrollo de un software es la de pruebas, la cual se realiza a lo largo del mismo, y constituye una vía de asegurar la detección y eliminación de los errores que aparezcan a lo largo del proceso, y por tanto de garantizar un adecuado funcionamiento del producto final.

Una de las sorpresas con las que se suelen encontrar los nuevos programadores es la enorme cantidad de tiempo y esfuerzo que requiere este flujo. Se estima que la mitad del esfuerzo de desarrollo de un programa (tanto en tiempo como en gastos) se va en este flujo.

Esta etapa es la encargada de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el proceso de desarrollo. El papel de las pruebas es planificar, ejecutar, controlar, evaluar, y proporcionar una realimentación a tiempo, de forma que las cuestiones de calidad puedan resolverse de manera efectiva en tiempo y costo.

…"La prueba de software involucra las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las condiciones controladas pueden ser normales o anormales. La prueba puede intencionalmente esforzar al programa y producir errores en las respuestas para determinar si los sucesos ocurren cuando no tendrían que ocurrir, o cuando los hechos no suceden cuando deberían suceder. La prueba de software está orientada a la detección. La mayoría de las grandes organizaciones asumen la responsabilidad del control de calidad y prueba de software a tal medida, que en la producción se incluyen desarrolladores de sistemas (analistas, programadores) y un grupo dedicado a la prueba de software para que estos grupos antes mencionados trabajen en conjunto cumpliendo el control de calidad (prevención) y la prueba de software (detección de errores) logrando una tarea exitosa"…. [Quesada, 2004].

Las actividades de este flujo comienzan desde el inicio del proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de las pruebas en el proyecto, así como las estrategias y recursos con que se dotará a esta

tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo de prueba consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida sirva para ir refinando el producto a desarrollar.

1.1 Definiciones de pruebas

La prueba de software es un conjunto de herramientas, técnicas y métodos que hacen la excelencia del desempeño de un programa. Las técnicas para encontrar problemas en un programa son extensamente variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad. Es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. Varios conceptos y definiciones le han sido dados a las pruebas de software, algunos de ellos se mencionan a continuación.

La prueba es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. [Gestión, 2006]

Prueba: “actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, los resultados importantes se observan, se registran y se realiza una evaluación de algún aspecto”. [IEEE, 1990].

Prueba es el proceso de ejecutar un programa con el fin de encontrar errores. El nombre de prueba además de la actividad de probar, se puede utilizar para designar un conjunto de casos y procedimientos de prueba. [Myers, 1979].

Prueba es cualquier intento de demostrar que el software tiene propiedades por debajo de la calidad requerida. [Torres, 2003]

De manera general las pruebas consisten en una serie de actividades, guiadas por técnicas y métodos, y en ocasiones realizadas a través de un conjunto de herramientas, que se llevan a cabo con el fin de encontrar los errores en el comportamiento de un sistema o componente del mismo. Los resultados se revisan, se registran y se hace una evaluación de los mismos.

1.2 Tendencias, Objetivos y Principios de las Pruebas

¿Cuál es la nueva tendencia en las pruebas?

- Iniciarlas antes, dentro del proyecto: las especificaciones de pruebas se realizan al mismo tiempo que el diseño de software; la propuesta es iniciar el análisis de las pruebas junto con el análisis del software. Ello habla de sondeos preventivos que permitan ejecutar las pruebas tan pronto como el software esté listo y con ello no sólo descubrir errores, sino evitarlos.
- Capacitar a especialistas responsables de esta actividad: crear conciencia acerca de la importancia de las pruebas y tener un equipo de personas dedicadas a esta actividad que puedan integrarse a un proyecto y sean responsables de su calidad.

Los objetivos actuales de las pruebas no sólo tienen que ver con corregir errores, sino con prevenirlos, influyendo y controlando el diseño y desarrollo del software. Las pruebas deben ser empleadas como modelos de los requerimientos de la aplicación que se ha de construir; por tanto, en las especificaciones de software deben incluirse especificaciones de pruebas, ambas deberán revisarse en conjunto, y en esta revisión deberá participar un especialista en pruebas. [Gurú, 2005]

Es frecuente encontrarse con el error de afirmar que el objetivo de este flujo de trabajo es convencerse de que el programa funciona bien. Cumplido ese objetivo, lo mejor posible, se pasa a llevar a cabo pruebas. Esto no es un obstáculo para reconocer que el objetivo último de todo el proceso de fabricación de programas sea hacer programas que funcionen bien; pero cada flujo tiene su objetivo específico, y el de las pruebas es destapar errores.

Probar un programa es ejercitarlo con la peor intención a fin de encontrarle fallos. Por poner un ejemplo duro, probar un programa es equivalente a la actividad de ciertos profesores para los que examinar a un alumno consiste en poner en evidencia todo lo

que no sabe. Esto es penoso cuando se aplica a personas; pero es exactamente lo que hay que hacerle a los programas. [Mañas, 1994]

De manera general, se puede decir que el objetivo central de las pruebas es diseñar pruebas que saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y espacio. [Pressman, 1998]

Otros objetivos de las pruebas pero más específicos son los que se exponen a continuación:

1. Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las de sistema, teniendo en cuenta una estimación inicial del tiempo, los recursos y el personal requerido para todo el proceso.
2. Diseñar e implementar las pruebas creando los casos de pruebas que especifican qué probar, creando los procedimientos de pruebas que especifican como realizar las pruebas y creando si es posible, componentes de prueba ejecutables para automatizar las pruebas.
3. 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. [Hetzel, 1988]

Antes de la aplicación de métodos para el diseño de casos de prueba efectivos, se deberá entender ciertos principios básicos que guían las pruebas del software. Se ha realizado una selección de los principios más importantes planteados por [Davis, 1995], [Myers, 1979]:

1. La prueba puede ser usada para mostrar la presencia de errores, pero nunca su ausencia. La principal dificultad del proceso de prueba es decidir cuando parar.
2. Evitar casos de pruebas no planificados, no reusables y triviales, a menos que el programa sea verdaderamente sencillo. Una parte necesaria de un caso de prueba es la definición del resultado esperado.

3. Los casos de pruebas tienen que ser escritos no solo para condiciones de entrada válidas y esperadas sino también para condiciones no válidas e inesperadas.
4. El número de errores sin descubrir es directamente proporcional al número de errores descubiertos.
5. Los pruebas deberían empezar por “lo pequeño“ y progresar hacia “lo grande“.
6. Todas las pruebas deberían ser capaces de hacer un seguimiento hasta los requisitos del cliente.

Otros principios de pruebas son los expuestos a continuación, planteados por [Kaner, 1993]:

1. Ha de tener una alta probabilidad de encontrar un fallo. Cuanto más, mejor. Para lograr esto, se debe entender el software e intentar desarrollar una imagen mental de cómo el mismo podría fallar.
2. Debe ser la “mejor de la cosecha”. Si se tiene donde elegir, se elige la mejor. Se debe emplear las pruebas que tengan las más alta probabilidad de descubrir una clase entera de errores.
3. No debería ser ni demasiado sencilla ni demasiado compleja. Si es muy sencilla no aporta nada, si es muy compleja a lo mejor no se sabe lo que ha fallado.

1.3 Facilidad de las Pruebas

La facilidad de las prueba a software es simplemente la facilidad con la que se puede probar un software. Como la prueba es profundamente difícil, merece la pena saber qué se puede hacer para hacerla más sencillo. La siguiente lista proporciona un conjunto de características o atributos que llevan a un software fácil de probar:

1. Operatividad: cuanto mejor funcione, más eficientemente se puede probar.

    - El sistema tiene pocos errores (los errores añaden sobrecarga de análisis y de generación de informes al proceso de pruebas).
    - Ningún error bloquea la ejecución de las pruebas.
    - El producto evoluciona en fases funcionales (permite simultanear el desarrollo y las pruebas).

2. Observabilidad: lo que ves es lo que pruebas

    - Se genera una salida distinta para cada entrada.
    - Los estados y variables del sistema están visibles o se pueden consultar durante la ejecución.
    - Todos los factores que afectan a los resultados están visibles.
    - Un resultado incorrecto se identifica fácilmente.
    - Los errores internos se detectan automáticamente a través de mecanismos de auto-comprobación.
    - Se informa automáticamente de los errores internos.
    - El código fuente es accesible.

3. Controlabibilidad: cuanto mejor se pueda controlar el software, más se puede automatizar y optimizar.

    - Todos los resultados posibles se pueden generar a través de alguna combinación de entrada.
    - Todo el código es ejecutable a través de alguna combinación de entrada.
    - El ingeniero de prueba puede controlar directamente los estados y las variables del hardware y el software.
    - Los formatos de las entradas y los resultados son consistentes y estructurados.
    - Las pruebas pueden especificarse, automatizarse y reproducirse convenientemente.

4. Capacidad de Descomposición: controlando el ámbito de las pruebas, se pueden aislar rápidamente los problemas y llevar a cabo mejores pruebas de regresión.

    - El sistema de software está construido con módulos independientes.
    - Los módulos del software se pueden probar independientemente.

5. Simplicidad: cuanto menos haya que probar, más rápidamente se podrá probarlo.

- Simplicidad funcional (el conjunto de características es el mínimo necesario para cumplir los requisitos).
- Simplicidad estructural (la arquitectura es modularizada para limitar la propagación de fallos).
- Simplicidad del código (se adopta un estándar de código para facilitar la inspección y el mantenimiento).

6. Estabilidad: cuanto menos cambio, menos interrupciones a las pruebas.

    - Los cambios del software ocurren con menor probabilidad.
    - Los cambios del software están controlados.
    - Los cambios del software no invalidan las pruebas existentes.
    - El software se recupera bien de los fallos.

7. Facilidad de Comprensión: cuanta más información se tenga, más inteligentes serán las pruebas.
    - El diseño se ha extendido perfectamente.
    - Las dependencias entre los componentes internos, externos y compartidos se han entendido perfectamente.
    - Se han facilitado los cambios del diseño.
    - La documentación técnica es instantáneamente accesible.
    - La documentación técnica está bien organizada.
    - La documentación técnica es específica y detallada.
    - La documentación técnica es exacta.

La facilidad de la prueba se usa para indicar cuan adecuadamente un conjunto particular de pruebas va a cubrir un producto. [Pressman, 2002]

1.4 Elementos Fundamentales de la Prueba

1.4.1 Métodos de Pruebas

Los métodos fundamentales de prueba son el método de caja negra y el de caja blanca. [Pressman, 2002]

La prueba de la caja blanca del software comprueba los caminos lógicos del software, proponiendo casos de prueba en los que se 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.

Se basan en el minucioso examen de los detalles procedimentales. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

Estas pruebas son métodos de casos de prueba que usan las propias estructuras de control del diseño procedural para derivar los casos de prueba; pudiéndose derivar casos de prueba que:

1. Garanticen que se ejercitan por lo menos una vez todos los caminos independientes de cada módulo.
2. Ejerciten todas las decisiones lógicas en sus caras verdadera y falsa.
3. Ejecuten todos los bucles en sus límites y con sus límites operacionales.
4. Empleen todas las estructuras de datos internas para asegurar su validez.

La realización de este tipo de pruebas se fundamenta en los siguientes puntos:

1. Los errores lógicos y las suposiciones incorrectas son inversamente proporcionales a la probabilidad de que se ejecute un camino del programa.
2. A menudo se cree que un camino lógico tiene pocas posibilidades de ejecutarse cuando, de hecho se puede ejecutar de forma regular.
3. Los errores tipográficos son aleatorios.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. [Ver Anexo 1]

Técnicas de Caja Blanca

Algunas técnicas de pruebas de Caja Blanca se exponen a continuación, tales como la Prueba del Camino Básico y las Pruebas de Bucles.

Prueba de Camino Básico

Este tipo de prueba de caja blanca permite derivar una medida de complejidad lógica de un diseño procedural y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución donde garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Este tipo de prueba se puede aplicar a un diseño procedural detallado o a un código fuente, y consta de los siguientes pasos [Pressman, 2002]:

1. Dibujar el grafo de flujo correspondiente a la aplicación (tomando como base el código o el diseño).
2. Determinar la complejidad ciclomática del grafo resultante. Es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Hay tres formas fundamentales de calcular la complejidad:

   - El número de regiones del grafo de flujo coincide con la complejidad ciclomática. La complejidad ciclomática, V(G), se define como:
     V (G) = A – N + 2
     Donde: A es el número de aristas del grafo y N es el número de nodos.
   - La complejidad ciclomática, V (G), también se define como:
     V (G) = P + 1
     Donde: P es el número de nodos predicado contenido en el grafo G.

3. Determinar un conjunto básico de caminos linealmente independientes. Es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.

   Si se diseñan pruebas que fuercen el recorrido de esos caminos, se garantiza que se ejecute al menos una vez cada sentencia del programa y que cada condición se ejecute en sus variantes verdadera y falsa. Se debe tener en

cuenta que de un mismo diseño procedimental se pueden derivar varios conjuntos básicos.

4. Preparar los casos de prueba. Se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

   Luego de confeccionar los casos de prueba se ejecutan cada uno de estos y se comparan los resultados con los esperados. Una vez terminados todos los casos de prueba, se estará seguro de que todas las sentencias del programa se han ejecutado por lo menos una vez. [Pressman, 2000]

Prueba de Bucles

Son la piedra angular de la inmensa mayoría de los algoritmos implementados en software. Este tipo de pruebas se basan en la validez de las construcciones de bucles. Se requiere un análisis de camino básico que aísle todos los posibles caminos del bucle, un conjunto especial de pruebas adicionales para cada tipo de bucle, en un intento de descubrir errores de inicialización, de indexación o de incremento y errores en los límites de los bucles. Los bucles se clasifican en simples, anidados y concatenados

Bucles simples: se les puede aplicar el siguiente conjunto de pruebas.

1. Saltar totalmente el bucle.
2. Pasar una sola vez por el bucle.
3. Pasar dos veces por el bucle.
4. Hacer *m*-pasos por el bucle $m < n$.
5. Hacer $n - 1$, n y $n + 1$ pasos por el bucle.

Bucles anidados: se propone el siguiente conjunto de pruebas:

1. Comenzar en el bucle más interno. Disponer todos los demás bucles en sus valores mínimos.

2. Llevar a cabo las pruebas de bucles simples para los bucles más internos dejando los externos con los iteradores en sus valores mínimos. Añadir otras pruebas para los valores fuera de rango o para valores excluidos.
3. Progresar hacia afuera, llevando a cabo las pruebas para el siguiente bucle, pero manteniendo todos los demás bucles exteriores en sus valores mínimos y los demás bucles anidados con valores típicos.
4. Continuar hasta que se hayan probado todos los bucles.

Bucles concatenados: se pueden probar mediante el enfoque anteriormente definido para los simples, siempre que estos sean independientes unos de otros, en otro caso se recomienda el enfoque de los bucles anidados.

Prueba de Caja Negra.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software donde los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Esta prueba examina algunos aspectos del modelo, fundamentalmente del sistema sin tener en cuenta la estructura interna del software. [Ver Anexo 2]. Los métodos de caja negra se centran en los requisitos funcionales del software, permitiendo al ingeniero del software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Este tipo de prueba ignora la estructura de control y se centra en el dominio de información. No suponen una alternativa a las pruebas de caja blanca, sino que permiten descubrir otra clase de errores:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y de terminación.

Las técnicas de prueba de caja negra permiten derivar un conjunto de casos de prueba que satisfacen los siguientes criterios:

1. Casos de prueba que reducen el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable.

Casos de prueba que dicen algo sobre la presencia o ausencia de clases de errores en lugar de un error asociado solamente con la prueba en particular que se encuentre disponible. [Pressman, 1998] [Myers, 1979] [Beizer, 1990]

Técnicas de Caja Negra

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas la técnica de Partición equivalente, Análisis de valores límites y la técnica de grafos causa-efecto.

Partición equivalente

Es un método que divide el dominio de entrada de un programa en clases de datos, de los que se pueden derivar casos de prueba. Se definen clases de equivalencia, para distintas condiciones de entrada, dirigidas a la definición de casos de prueba que descubran clases de errores. Las clases de equivalencia se pueden definir con arreglo a las siguientes directrices:

1. Si una condición de entrada especifica un rango, entonces se definen una clase de equivalencia válida y dos inválidas.
2. Si una condición de entrada requiere un valor específico, entonces se definen una clase de equivalencia válida y dos inválidas.
3. Si una condición de entrada especifica un miembro de un conjunto, entonces se definen una clase de equivalencia válida y una inválida.
4. Si una condición de entrada es booleana, entonces se definen una clase de equivalencia válida y otra inválida.

Análisis de valores límite

Los errores tienden a darse más en los límites del dominio de datos que en el interior. Es por tanto natural el desarrollar un método de prueba centrada en estos valores límites que conducen a un mayor número de errores.

El análisis de valores límite se centra en generar casos de prueba para probar la corrección de las condiciones límites, que son aquellas situaciones que se dan cuando se introducen valores que están justo en el límite de las clases de equivalencia del

dominio, tanto de entrada como de salida. Es un método que complementa la partición equivalente y que presenta notables diferencias con éste:

1. No se selecciona un elemento representativo de una clase de equivalencia sino uno o más, tal que se prueben los extremos de dicha clase.
2. Se presta atención no sólo al domino de entrada sino también al dominio de salida.

Una ventaja considerable de este método es que presenta una mayor probabilidad de detectar errores no detectados hasta entonces.

Criterios para la derivación de casos de prueba

1. Si una condición de entrada especifica un rango de valores limitado por a y b, se deben escribir casos de prueba para los límites del rango, es decir, a y b, y casos de prueba inválidos para situaciones fuera de los límites. Ej: si el rango válido de un valor de entrada es -1.0 a +1.0, escribir casos de prueba para las situaciones: -1.0, +1.0, -1.0001, +1.0001
2. Si una condición de entrada especifica un número de valores, desarrollar casos de prueba para el número máximo y mínimo de valores y otros casos para los valores justo por encima y justo por debajo del máximo y del mínimo. Ej: si el número de alumnos para calcular su nota final es de 125, escribir casos de prueba para 0, 1, 125 y 126 alumnos.
3. Aplicar el primer criterio para cada condición de salida. Por ejemplo, si un programa calcula la deducción fiscal mensual, siendo el rango permitido entre 0 y 1000 €. escribir casos de prueba que proporcionen el resultado de 0 € y 1000 €. Comprobar si es posible diseñar casos de prueba que puedan proporcionar una deducción negativa o mayor de 1000 €.
4. Es importante examinar los límites del espacio de resultados pues no siempre ocurre que los límites de los dominios de entrada representan el mismo conjunto de circunstancias que los límites de los rangos de salida. (por ejemplo la función seno).
5. Usar el segundo criterio para cada condición de salida. Por ejemplo, si el número máximo de errores de tipo léxico que detecta un analizador léxico es de 10, escribir casos de prueba para que el analizador no saque ningún error, un error, diez errores, e intentar que el analizador produzca once errores.
6. Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar casos de prueba que ejerciten la estructura de datos en

sus límites. Por ejemplo, un array que tenga un límite definido de 100 elementos, habrá que probar el hecho de que no existan elementos y el de que se procesen bien 100 elementos.

7. Usar el ingenio para buscar otras condiciones límites.

Técnicas de grafos causa-efecto

Es una técnica de diseño de casos de prueba que proporciona una concisa representación de las condiciones lógicas y sus correspondientes acciones. La técnica sigue cuatro pasos:

1. Se listan para un módulo las causas (condiciones de entrada) y los efectos (acciones), asignando un identificador a cada uno de ellos.
2. Se desarrolla un grafo de causa-efecto.
3. Se convierte el grafo en una tabla de decisión.
4. Se convierten las reglas de la tabla de decisión a casos de prueba, al menos una prueba por cada regla. [Pressman, 2002]

1.4.2 Niveles de Prueba

Las pruebas son aplicadas a diversos objetos de pruebas, en diferentes etapas o niveles del esfuerzo de trabajo. Estos niveles son típicamente distinguidos por esos roles que mejor capacidad tienen para diseñar y conducir las pruebas, y donde las técnicas son más apropiadas para probar en cada nivel. Es importante garantizar un equilibrio en el enfoque que se mantenga para estos diferentes esfuerzos de trabajo.

Existen diferentes niveles de pruebas, cada una de ellas se realizan en determinados momentos del ciclo de vida del software. Según [Pérez, 2006] se distinguen los siguientes niveles de pruebas como:

Prueba de Unidad: Es el proceso de verificación en la menor unidad del diseño del software: el módulo, se prueban los caminos de control importantes, con el fin de descubrir errores dentro del límite del módulo. La prueba de unidad siempre está orientada a caja blanca. Su objetivo es probar el comportamiento de cada uno de los componentes de forma independiente. Este tipo de prueba deberá realizarse una vez que sea implementado el componente. Se prueba la funcionalidad de una clase o conjunto de clases correlacionadas.

Prueba de Integración: Es el proceso de combinar y probar múltiples componentes juntos. El objetivo es construir una estructura de programa, a partir de los módulos probados en el nivel de unidad, que esté de acuerdo con lo que dicta el diseño. Una vez que se probaron los componentes individuales del programa, deben integrarse para crear un sistema parcial, o completo. Este proceso de integración comprende la construcción del sistema y las pruebas al sistema resultante con respecto a los problemas que surjan de las iteraciones de los componentes. Las pruebas de integración se desarrollan a partir de las especificaciones de los componentes y de la especificación del sistema y son iniciadas tan pronto como estén disponibles versiones utilizables de algunos componentes del sistema. Las principales dificultades que surgen en la pruebas de integración es localizar los errores que se descubren durante el proceso. Existen interacciones complejas entre los componentes del sistema y cuando se descubre una salida anómala, es difícil encontrar la fuente del error. Para hacer más fácil la localización de errores, siempre se utiliza un enfoque incremental para la integración y pruebas del sistema. De forma inicial, se debe integrar una configuración mínima del sistema y probar dicho sistema. Luego se agrega componentes a esta configuración mínima y se prueba después de que se agrega cada incremento.

Prueba de Sistema: Es el proceso de integrar el sistema y el hardware para verificar que el sistema reúna los requerimientos especificados. Una prioridad de este sistema es que debe concentrarse más en la capacidad del sistema que en la capacidad de los componentes, es decir en el uso de la interacción de las funciones que en probar los detalles de implementación, otra prioridad es probar las situaciones críticas en lugar de probar casos especiales. La persona encargada de hacer las pruebas debe encontrar las áreas en donde los módulos hayan sido diseñados con especificaciones distintas para la longitud y tipo de datos y los nombres de los elementos de dato. Es la combinación sistemática con una ejecución de componentes de subsistemas.

Prueba de Implantación: Su objetivo es la comprobación de cualquier detalle de diseño interno hasta aspectos como las comunicaciones. Se deben verificar fundamentalmente los requisitos no funcionales definidos para el producto como por ejemplo, comprobar que el sistema puede gestionar los volúmenes de información requeridos, que el sistema se ajusta a los tiempos de respuesta deseados y que los procedimientos de respaldo, seguridad e interfaces con otros sistemas funcionan

correctamente. Se debe verificar también el comportamiento del sistema bajo las condiciones más extremas.

Pruebas de Aceptación: son las pruebas finales, antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado eficientemente por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. Estas pruebas son realizadas por el cliente y usuario final.

Prueba Alfa

Es la prueba que se realiza cuando se está cerca de la entrega al usuario de la aplicación Se hacen pequeños cambios generalmente en el diseño de interfaces. Esta prueba es hecha por usuarios en el lugar de desarrollo del producto, donde participa el desarrollador como observador.

Prueba Beta

Es la búsqueda de bugs en el programa completo. Generalmente es hecha por usuarios finales, en el lugar de trabajo de estos, y el desarrollador generalmente no está presente.

Pruebas de Regresión: es un conjunto de pruebas que se ejecutan nuevamente. Su objetivo es verificar que cambios ocurridos en un componente del sistema no provoca errores adicionales en otros componentes del mismo que ya han sido probados. Asegura que los cambios no introducen un comportamiento no deseado o errores adicionales.

1.4.3 Tipos de Pruebas

Hoy en día, la mayor parte de las técnicas de prueba le dan fundamental importancia a los avances tecnológicos, los avances en los lenguajes de programación y la gran variedad de tipos de software. Las pruebas de caja negra y caja blanca han tomado un lugar muy importante en el desarrollo de sistemas de cualquier tipo, tanto que sin dichas pruebas un sistema desarrollado carecería de garantías y credibilidad en sus resultados. Pero además, se desarrollan una serie de pruebas para verificar el comportamiento del sistema. Las técnicas de pruebas funcionales son efectivas en detectar falla en todos los niveles de pruebas, pero deben ser usadas en combinación con otras para hacerlas más efectivas. Estas técnicas han de dar cumplimiento a una

serie de parámetros relativos a la dimensión de la calidad del software, y al tratamiento de los riesgos.

Dimensión de calidad/ Dimensión de riesgo	Tipo de pruebas	Características
Funcionalidad	Pruebas Funcionales	Se enfocan en validar que los elementos que se están probando funcionan según el propósito para el cual fueron creados, de manera que provean los servicios requeridos, métodos o casos de uso. Estas pruebas son implementadas y ejecutadas de acuerdo a los elementos a probar definidos, incluyendo unidades, unidades integradas, aplicaciones y sistemas.
	Pruebas de Seguridad	Evalúan que tan bien el sistema se protege contra accesos, internos o externos, no autorizados, esta prueba requiere sofisticadas técnicas y herramientas.
	Pruebas de Volumen	Estas pruebas están enfocadas a verificar la habilidad de los elementos a probar de manejar grandes cantidades de datos, ya sean datos de entrada o salida de la base de datos. Estas pruebas incluyen estrategias, tales como crear consultas que podrían devolver todo el contenido de la base de datos, o que podrían tener tantas restricciones que no se devuelva ningún dato, o donde los

		datos de entrada tienen la máxima cantidad de datos por cada campo.
Usabilidad	Pruebas de Usabilidad	Se enfocan en: • Factores humanos • Estética • Consistencia en la interfaz de usuario • Ayuda online y ayuda de contenido sensible • Documentación de usuario • Materiales de entrenamiento
Fiabilidad	Pruebas de Integridad	Pruebas enfocadas en la evaluación de la robustez (resistencia a fallas) de los blancos de pruebas, y ajuste técnico al lenguaje, sintaxis y uso de los recursos. Esta prueba es implementada y ejecutada a varios blancos de pruebas, incluyendo unidades y unidades integradas.
	Pruebas de estructura	Pruebas enfocadas en la evaluación de adhesión de los blancos de pruebas a su diseño y formación. Típicamente, esta prueba es hecha para aplicaciones Web, asegurando que se conecten todos los vínculos, que se desplieguen los contenidos apropiados y que ningún contenido quede huérfano, o sea, que no haya forma acceder o de presentar dicho contenido.
	Pruebas de Stress	Se enfocan en evaluar como el sistema responde en condiciones

		anormales. El stress en el sistema puede incluir carga de trabajo extrema, memoria insuficiente, servicios de hardware no disponibles, o recursos compartidos limitados. Estas pruebas con frecuencia son realizadas para lograr un mejor entendimiento de cómo y en que áreas el sistema puede fallar, de manera que los planes de contingencia y las actualizaciones de mantenimiento puedan ser planificadas y presupuestadas para bien en lo adelante.
Rendimiento	Pruebas de Referencia	Comparan el rendimiento de un elemento a probar nuevo o desconocido, con a un referencia conocida de carga de trabajo y sistema
	Pruebas de Contención	Pruebas enfocadas en la validación de las habilidades del blanco de pruebas para manipular aceptablemente demandas de múltiples actores en el mismo recurso (registro de datos, memoria, etc.)
	Pruebas de Carga	Se utilizan para validar y evaluar aceptablemente los límites operacionales de un sistema bajo cargas de trabajo variables, mientras que el sistema bajo prueba permanece constante y la configuración de sistema bajo prueba es variada también, se determina en cuales puntos existen degradaciones del sistema.
	Pruebas de Perfil del Rendimiento	Se monitorea el perfil de tiempo, incluyendo el flujo de ejecución, los

		accesos a datos, las funciones y las llamadas al sistema para identificar y direccionar los procesos ineficientes y los problemas de rendimiento.
Soporte	Pruebas de Configuración	Se enfocan en asegurar que los elementos a probar cumplan sus funciones en diversas configuraciones de software y hardware. Esta prueba también puede ser implementada como una prueba de rendimiento del sistema.
	Pruebas de instalación	Se enfocan en asegurar que loe elementos a probar se puedan instalar en diversas configuraciones de hardware y software, y bajo diferentes condiciones (tales como espacio en el disco insuficiente o interrupciones en el fluido). Estas pruebas son implementadas y ejecutadas según las aplicaciones y los sistemas.

Tabla 1 Tipos de Pruebas de Software. [RUP, 2003]

Otros tipos de pruebas a software que existen según [Mañas, 1994], [Teruel, 2001] son:

Prueba de Exploración

Es una prueba informal del software que no está basada en ningún plan o caja de prueba y a menudo los probadores aprenden del programa al explorar todas las aplicaciones posibles.

Prueba de Anuncio

Es similar a la prueba de exploración pero los probadores deben tener suficiente noción sobre el funcionamiento del programa antes de comenzar esta prueba. Incluye reunión con analistas y programadores.

Prueba de Usuario

Determina si el usuario se desenvuelve satisfactoriamente con el programa.

Prueba de Comparación

En esta prueba se comparan los pro y los contra del programa con los programas creados con la competencia.

Prueba de sanidad

Determina si la nueva versión de un software está bien realizada y si necesita un nuevo esfuerzo en la prueba de software. Por ejemplo la nueva versión de un programa cumple con casi todos los requisitos pero destruye la base de datos al leerla, por lo tanto se dice que este software no está en una condición sana.

Recorridos

Quizás es una técnica más aplicada en control de calidad que en pruebas. Consiste en sentar alrededor de una mesa a los desarrolladores y a una serie de críticos, bajo las órdenes de un moderador que impida un recalentamiento de los ánimos. El método consiste en que los revisores se leen el programa línea a línea y piden explicaciones de todo lo que no está medianamente claro. Puede que simplemente falte un comentario explicativo, o que detecten un error auténtico o que simplemente el código sea tan complejo de entender/explicar que más vale que se rehaga de forma más simple. Para un sistema complejo pueden hacer falta muchas sesiones. Esta técnica es muy eficaz localizando errores de naturaleza local; pero falla estrepitosamente cuando el error se deriva de la interacción entre dos partes alejadas del programa. Nótese que no se está ejecutando el programa, sólo mirándolo con lupa, y de esta forma sólo se ve en cada instante un trocito del listado.

Pruebas Aleatorias

Ciertos autores consideran injustificada una aproximación sistemática a las pruebas. Alegan que la probabilidad de descubrir un error es prácticamente la misma si se hacen una serie de pruebas aleatoriamente elegidas, que si se hacen siguiendo las instrucciones dictadas por criterios de cobertura (caja negra o blanca).

Como esto es muy cierto, probablemente sea muy razonable comenzar la fase de pruebas con una serie de casos elegidos al azar. Esto pondrá de manifiesto los errores más patentes. No obstante, pueden permanecer ocultos errores más sibilinos que sólo se muestran ante entradas muy precisas. Si el programa es poco crítico (una aplicación personal, un juego,...) puede que esto sea suficiente. Pero si se trata de una aplicación militar o con riesgo para vidas humanas, es de todo punto insuficiente.

Solidez

Se prueba la capacidad del sistema para salir de situaciones embarazosas provocadas por errores en el suministro de datos. Estas pruebas son importantes en sistemas con una interfaz al exterior, en particular cuando la interfaz es humana. Por ejemplo, en un sistema que admite una serie de órdenes (commands) se deben probar los siguientes extremos:

- órdenes correctas, todas y cada una
- órdenes con defectos de sintaxis, tanto pequeñas desviaciones como errores de bulto
- órdenes correctas, pero en orden incorrecto, o fuera de lugar
- la orden nula (línea vacía, una o más)
- órdenes correctas, pero con datos de más
- provocar una interrupción (BREAK, ^C, o lo que corresponda al sistema soporte) justo después de introducir una orden.
- órdenes con delimitadores inapropiados (comas, puntos, ...)
- órdenes con delimitadores incongruentes consigo mismos (por ejemplo, esto)

Conformidad u Homologación

En programas de comunicaciones es muy frecuente que, además de los requisitos específicos del programa que se está construyendo, aparezca alguna norma más amplia a la que el programa deba atenerse. Es frecuente que organismos internacionales como ISO y el CCITT elaboren especificaciones de referencia a las que los diversos fabricantes deben atenerse para que sus ordenadores sean capaces de entenderse entre sí. Las pruebas de caja negra, que se le pasan a un producto para detectar discrepancias respecto a una norma de las descritas en el párrafo anterior se denominan de conformidad u homologación. Suelen realizarse en un centro especialmente acreditado al efecto y, si se pasan satisfactoriamente, el producto recibe un sello oficial que dice: "homologado".

Mutación

Esta prueba está basada en la introducción deliberada de diferentes códigos externos al programa (bugs) para reexaminar si estos bugs pueden ser detectados. Requiere gran disponibilidad de recursos de computación.

Es una técnica curiosa consistente en alterar ligeramente el sistema bajo pruebas (introduciendo errores) para averiguar si la batería de pruebas es capaz de detectarlo. Si no, más vale introducir nuevas pruebas. Todo esto es muy laborioso y francamente artesanal.

Depuración

Casi todos los compiladores suelen llevar asociada la posibilidad de ejecutar un programa paso a paso, permitiéndole al operador conocer dónde está en cada momento, y que valor tienen las variables. Los depuradores pueden usarse para realizar inspecciones rigurosas sobre el comportamiento dinámico de los programas. La práctica demuestra, no obstante, que su uso es tedioso y que sólo son eficaces si se persigue un objetivo muy claro. El objetivo habitual es utilizarlo como consecuencia de la detección de un error. Si el programa se comporta mal en un cierto punto, hay que averiguar la causa precisa para poder repararlo. La causa a veces es inmediata (por ejemplo, un operador booleano equivocado); pero a veces depende del valor concreto de los datos en un cierto punto y hay que buscar la causa en otra zona del programa. En general es mala idea "correr al depurador", tanto por el tiempo que se pierde buceando sin una meta clara, como por el riesgo de corregir defectos intermedios sin llegar a la raíz del problema. Antes de entrar en el depurador hay que delimitar el error y sus posibles causas. Ante una prueba que falla, hay que identificar el dominio del fallo, averiguar las características de los datos que provoca el fallo (y comprobar experimentalmente que todos los datos con esas características provocan ese fallo, y los que no las tienen no lo provocan).

Prueba de almacenamiento

Los analistas cuando diseñan el sistema, deben analizar también el espacio que puede llegar a ocupar, funcionando a los límites, buscando cierta compatibilidad con el espacio de Disco que se va a utilizar en la computadora que se va a correr. Justamente esto se tiene que probar antes de la puesta en marcha.

Prueba de Recuperación

Sólo con crear una situación de pérdida ó de falla de datos donde los usuarios se vean obligados a volver a cargar y recuperar una copia de respaldo, los analistas pueden determinar si los procedimientos de recuperación son adecuados.

Prueba de procedimiento

El analista pone siempre mucho énfasis en la creación de los manuales de procedimiento, ya que un buen diseño de ellos es insustituible, sobre todo siendo lo más críticos posible en los detalles del manejo del sistema incluyendo todo esto en la documentación.

Lo que en realidad el analista necesita es ejercer la prueba de Recursos Humanos. Donde se observa en el usuario las acciones y reacciones que va teniendo en el manejo del sistema. Cuidarse de pantallas en blanco donde quizás el usuario apague, etc., tratar de poner siempre carteles si hay algún proceso para que el usuario se informe. En resumen, es responsabilidad del analista anticipar las preguntas que surgirán en la mente de los usuarios cuando interactúan con el sistema. Otra prueba importante es ver como el usuario carga los datos en el sistema.

1.4.4 Artefactos de Pruebas

1.4.4.1 Casos de pruebas

El propósito de un caso de prueba es especificar una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, los resultados esperados y las condiciones bajo las que ha de probarse. Los casos de prueba son un producto de desarrollo de software que ayudan a validar y verificar las expectativas de los involucrados.

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular ó una función esperada. La entidad más simple siempre es ejecutada como una unidad, desde el comienzo hasta el final, estos casos de pruebas se debe verificar:

1. Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.
2. Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

Formalmente, los casos de prueba están descritos en documentos formales que consisten principalmente en tres partes con subdivisiones:

1. Introducción/visión general contiene información general acerca los casos de prueba.

- Identificador: es un identificador único para futuras referencias, por ejemplo, mientras se describe un defecto encontrado.
- Caso de prueba dueño/creador: es el nombre del analista o diseñador de pruebas, quien ha desarrollado pruebas o es responsable de su desarrollo.
- Versión: la actual definición del caso de prueba.
- Nombre del caso de prueba: debe ser un título entendible por personas, para la fácil comprensión del propósito del caso de prueba y su campo de aplicación.
- Identificador de requerimientos: está incluido por el caso de prueba. También aquí puede ser identificador de casos de uso o especificación funcional.
- Propósito: contiene una breve descripción del propósito de la prueba, y la funcionalidad que chequea.

2. Actividad de los casos de prueba

- Ambiente de prueba/configuración: contiene información acerca del la configuración del hardware o software en el cuál se ejecutará el caso de prueba.
- Inicialización: describe acciones que deben ser ejecutadas antes de que los casos de prueba se hayan inicializado. Por ejemplo, se debe abrir algún archivo.
- Finalización: describe acciones que deben ser ejecutadas después de realizado el caso de prueba. Por ejemplo, si el caso de prueba estropea la base de datos, el analista debe restaurarla antes de que otro caso de prueba sea ejecutado.
- Acciones: pasos a realizar para completar la prueba.
- Descripción: se describen los datos de entrada

3. Resultados

- Resultados esperados: contiene una descripción de lo que el analista debería ver tras haber completado todos los pasos de la prueba
- Resultados reales: contienen una breve descripción de lo que el analista encuentra después de que los pasos de prueba se hayan completado. Esto se sustituye a menudo con un Correcto/Fallido. Si un

caso de prueba falla, frecuentemente la referencia al defecto implicado se debe enumerar en esta columna.

En la práctica lo que se prueba puede venir dado por un requisito o una colección de requisitos del sistema, cuya implementación justifica una prueba que es posible realizar y que llevarla a cabo no resultada demasiado caro.

El diseño de casos de prueba está totalmente mediatizado por la imposibilidad de probar exhaustivamente el software. En consecuencia, las técnicas de diseño de casos de prueba tienen como objetivo conseguir una confianza aceptable en que se detectarán los defectos existentes (ya que la seguridad total sólo puede obtenerse de la prueba exhaustiva, que no es practicable) sin necesidad de consumir una cantidad excesiva de recursos. Toda la disciplina de pruebas debe moverse, por lo tanto, en un equilibrio entre la disponibilidad de recursos y la confianza que aportan los casos de pruebas para descubrir los defectos existentes. Este equilibrio no es sencillo, lo que convierte a las pruebas en una disciplina difícil que está lejos de parecerse a la imagen de actividad rutinaria que suele sugerir. Ya que no se pueden probar todas las posibilidades de funcionamiento del software, la idea fundamental para el diseño de casos de prueba consiste en elegir algunas de ellas que, por sus características, se consideren representativas del resto. Así, se asume que, si no se detectan defectos en el software al ejecutar dichos casos, se puede tener un cierto nivel de confianza en que el programa no tiene defectos. La dificultad de esta idea es saber elegir los casos que se deben ejecutar. [Teruel, 2001]

1.4.4.2 Diseño de Datos de prueba

- Utilización de datos reales de prueba.

Estos datos son los que se extraen de los archivos de la empresa. Los analistas les piden a los usuarios que carguen datos como si estuvieran en realidad realizando sus actividades normales en la empresa. Conseguir datos reales suficientes no siempre se logra y a veces no se ven todas las combinaciones o formatos que pueden llegar a introducirse. Lamentablemente, se llega a ignorar en algunos casos, los datos que realmente generan fallas en el sistema.

- Utilización de datos artificiales de prueba.

Estos datos se crean únicamente para propósitos de prueba, dado que se arman para probar todo aquello que con los datos reales no se pudo. Hacen posible la prueba de todas las rutas lógicas y de control a través del programa. [Soto, 2006]

1.4.4.3 Bibliotecas de prueba

Es un conjunto de datos desarrollados para probar completamente un sistema de programas, se almacenan en forma legible en Disco y se utilizan por todas las personas que trabajan con un sistema específico. Estas bibliotecas sirven también para probar el sistema cuando ya está avanzado, como un mantenimiento mientras el sistema va evolucionando y los programas se van modificando y se deben volver a probar. Por la tanto las Bibliotecas de datos se deben mantener a través de la vida del sistema, de manera que conforme se realiza cada cambio se tengan disponibles otra vez datos confiables para probar el sistema. [Prado, 2006]

1.4.4.4 Repositorio de pruebas

Es un lugar de almacenamiento de los artefactos de salida que se generan durante el proceso de pruebas, tales como el plan de pruebas, la lista de no conformidades, la lista de defectos, etc.

1.4.4.5 Plan de Pruebas

La construcción de un buen Plan de Pruebas es la piedra angular y en consecuencia el principal factor crítico de éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel. No obstante que cada esfuerzo o proceso de pruebas puede ser diferente y específico, la mayor parte de los proyectos informáticos, sean de nuevos desarrollos o de mantenimiento de aplicaciones, tienen un marco común para la realización de las pruebas.

Según la norma IEEE 829-1983, el propósito del plan de pruebas es dejar de forma explicita el alcance, el enfoque, los recursos requeridos, el calendario, los responsables y el manejo de riesgos de un proceso de pruebas, así como los elementos a probar, las características, las actividades de prueba, el personal responsable y los riesgos. [Ver Anexo 3]
Describe las estrategias, recursos y planificación de la prueba. La estrategia de prueba incluye la definición del tipo de pruebas a realizar para cada iteración y sus objetivos, el nivel de cobertura de prueba y de código necesario y el porcentaje de pruebas que deberían ejecutarse con un resultado específico.
Está constituido por un conjunto de pruebas donde cada prueba debe:

1. Dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad).

2. Dejar claro cómo se mide el resultado.
3. Especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta).
4. Definir cual es el resultado que se espera (identificación, tolerancia). ¿Cómo se decide que el resultado es acorde con lo esperado?

Un plan de prueba incluye:

1. Identificador del plan: preferiblemente de alguna forma nemónica que permita relacionarlo con su alcance, por ej.TP-Global (plan global del proceso de pruebas), TP-Req-Seguridad1 (plan de verificación del requerimiento 1 de seguridad), TP-Contr-X (plan de verificación del contrato asociado al evento de sistema X), TP-Unit-Despachador.iniciar (plan de prueba unitario para el método iniciar de la clase Despachador). Como todo artefacto del desarrollo, está sujeto a control de configuración, por lo que debe distinguirse adicionalmente la versión y fecha del plan.
2. Alcance: indica el tipo de prueba y las propiedades o elementos del software a ser probado.
3. Ítems a probar: indica la configuración a probar y las condiciones mínimas que debe cumplir para comenzar a aplicarle el plan. Por un lado, es difícil y riesgoso probar una configuración que aún reporta fallas; por otro lado, si se espera a que todos los módulos estén perfectos, puede que se detecten fallas graves demasiado tarde.
4. Estrategia: describe la técnica, patrón y/o herramientas a utilizarse en el diseño de los casos de prueba. Por ejemplo, en el caso de pruebas unitarias de un procedimiento, esta sección podría indicar: "Se aplicará la estrategia caja-negra de fronteras de la precondición" o "Ejercicio de los caminos ciclomáticos válidos". En lo posible la estrategia debe precisar el número mínimo de casos de prueba a diseñar, por ej. 100% de las fronteras, 60% de los caminos ciclomáticos. La estrategia también explicita el grado de automatización que se exigirá, tanto para la generación de casos de prueba como para su ejecución.
5. Categorización de la configuración: explicita las condiciones bajo las cuales, el plan debe ser: Suspendido, Repetido, Culminado. En algunas circunstancias (las cuales deben ser explicitadas) el proceso de prueba debe suspenderse en vista de los defectos o fallas que se han detectado. Al corregirse los defectos, el proceso de prueba previsto por el plan puede continuar, pero debe explicitarse a partir de qué punto, ya que puede ser necesario repetir algunas

pruebas. Los criterios de culminación pueden ser tan simples como aprobar el número mínimo de casos de prueba diseñados o tan complejos como tomar en cuenta no sólo el número mínimo, sino también el tiempo previsto para las pruebas y la tasa de detección de fallas.

6. Tangibles: define los documentos a entregarse al culminar el proceso previsto por el plan p. ej. subplanes, especificación de pruebas, casos de prueba, resumen gerencial del proceso y bitácora de pruebas.
7. Procedimientos especiales: identifica el grafo de las tareas necesarias para preparar y ejecutar las pruebas, así como cualquier habilidad especial que se requiere.
8. Recursos: especifica las propiedades necesarias y deseables del ambiente de prueba, incluyendo las características del hardware, el software de sistemas (p. ej. el sistema de operación), cualquier otro software necesario para llevar a cabo las pruebas, así como la colocación específica del software a probar (p. ej. qué módulos se colocan en qué máquinas de una red local) y la configuración del software de apoyo.
9. Calendario: Esta sección describe los hitos del proceso de prueba y el grafo de dependencia en el tiempo de las tareas a realizar.
10. Manejo de riesgos: establece los riesgos del plan, las acciones mitigantes y de contingencia.
11. Responsables: especifica quién es el responsable de cada una de las tareas previstas en el plan. [Teruel, 2001]

1.4.4.6 Estrategia de pruebas

Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a una construcción correcta del software. Estas estrategias de pruebas presentan un conjunto de características generales, como son:

- La prueba comienza en el nivel de módulo y trabaja "hacia fuera", hacia la integración completa del sistema completo.
- En diferentes puntos es adecuada la utilización de técnicas de prueba distintas.
- La prueba la lleva a cabo el que desarrolla el software y (para grandes proyectos) un grupo de prueba independiente.
- La prueba y la depuración son actividades diferentes, pero la depuración puede entrar en cualquier estrategia de prueba.

Una estrategia de prueba del software proporciona un mapa para la organización de control de calidad y del cliente, que debe seguir el responsable de desarrollo. Dicho mapa describe los pasos que se deben seguir como parte de la prueba, cuándo se debe planificar y cuanto esfuerzo, tiempo y recursos es necesario emplear. Cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los resultados. [Moreno, 2003]

1.6.4.7 Procedimientos de Pruebas

Un procedimiento de prueba especifica como realizar uno o varios casos de prueba o parte de estos. Por ejemplo un procedimiento de prueba puede ser una instrucción para un individuo sobre como ha de realizar un caso de prueba manualmente, o puede ser una especificaron de cómo interaccionar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables de pruebas.
El como llevar a cabo un caso de prueba puede ser especificado por un procedimiento de prueba pero es a menudo útil reutilizar un procedimiento de prueba para varios casos de prueba y reutilizar varios procedimientos de prueba para varios casos de prueba.
Los diseñadores de pruebas intentan crear procedimientos que puedan ser reutilizables en varios casos de prueba, esto permite usar un conjunto reducido de procedimientos de pruebas con rapidez y precisión para muchos casos de prueba.
Cada caso de prueba precisa varios procedimientos de prueba, quizás uno por cada subsistema de servicio probado en el caso de pruebas. Relacionando de esta forma los procedimientos de prueba con los subsistemas de servicios los procedimientos serian más fáciles de mantener. [RUP, 2003]

1.5 Modelos de Pruebas de Software

Un proceso de pruebas constituye un grupo de actividades y entregables que se siguen para realizar pruebas a un software garantizando un producto confiable, sin errores y con la mayor calidad posible. Cada prueba de software que se realice debe ser planificada, diseñada, ejecutada, controlada, y se le debe dar un seguimiento para lograr una mejora continua del proceso que se lleva a cabo.

El proceso de prueba comienza con la generación de un plan de pruebas sobre la base de la documentación del proyecto y la documentación del software a probar. A partir de dicho plan, se entra en detalle diseñando pruebas específicas, basándose en la documentación del software a probar. Una vez detalladas las pruebas (especificaciones de casos y de procedimientos de pruebas), se toma la configuración del software (revisada, para confirmar que se trata de la versión apropiada del programa) que se va a probar para ejecutar sobre ella los casos de pruebas. En algunas situaciones, se puede tratar de reejecuciones de pruebas, por lo que es conveniente tener constancia de los defectos ya detectados aunque aún no corregidos. A partir de los resultados de salida, se pasa a su evaluación mediante una comparación con la salida esperada. A partir de ésta, se pueden realizar dos actividades:

- La depuración (localización y corrección de defectos).
- El análisis de la estadística de errores.

La depuración puede corregir o no los defectos. Si no se consigue localizarlos, puede ser necesario realizar pruebas adicionales para obtener más información. Si se corrige un defecto, se debe volver a probar el software para comprobar que el problema está resuelto. Por su parte, el análisis de errores puede servir para realizar predicciones de la fiabilidad del software y para detectar las causas más habituales de error y mejorar los procesos de desarrollo.

El proceso de prueba generalmente recibe muy poca atención en los modelos que se presentan a continuación y usualmente aparece como una tarea poco atractiva para llevar a cabo después de la implementación.

1.5.1 Modelo V

Este modelo describe, a un nivel alto de abstracción, las fases del ciclo de desarrollo en las que se involucran las pruebas y los niveles de las mismas.

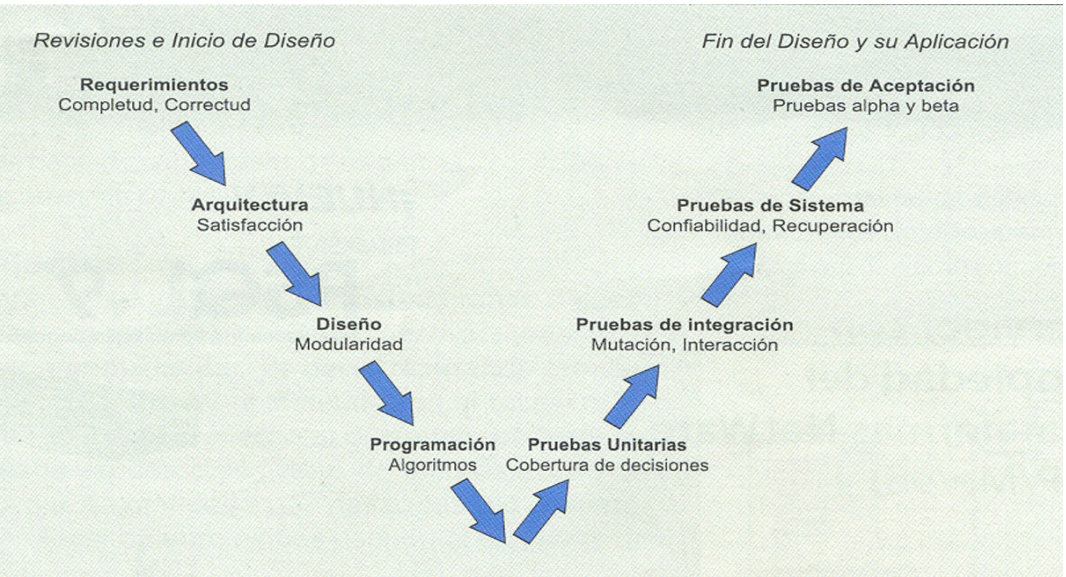

Fig. 1 Modelo de Prueba V [GURU, 2005]

La prueba comienza haciendo revisiones técnicas a los requerimientos y haciendo un esfuerzo de los primeros casos de pruebas de aceptación, se continua revisando que la arquitectura satisfaga los requerimientos y definiendo los primeros casos de pruebas de sistemas. Después se revisa la modularidad del diseño y se esfuerzan los casos de pruebas de integración, para luego pasar a revisar los algoritmos y a desarrollar los casos de pruebas de unidad.

Las actividades de pruebas de la línea izquierda de la V se llevan a cabo en paralelo al desarrollo de software y se produce de cierta forma la documentación y la descripción de las pruebas. La línea de la derecha involucra la terminación del diseño de los casos de pruebas y la aplicación de los mismos, en esta parte se producen los informes resultantes de las pruebas.

La V es también un sinónimo para la verificación y validación. Este modelo es muy simple y fácil de entender. Por el orden de las actividades en secuencia de tiempo y con la abstracción de niveles la conexión entre desarrollo y actividades de pruebas se vuelven más claras.

Así, por ejemplo, el plan de pruebas y la estrategia de prueba podrían ser definidos inmediatamente después de la definición de los requerimientos. No obstante, esto podría perfectamente contribuir a la estructuración del proceso de desarrollo de software.

La desventaja de este modelo radica en la basta división en el trabajo de construcción (incluyendo la implementación) en el lado izquierdo de la V y las tareas más destructivas en el lado derecho de la V. [GURU, 2005]

1.5.2 Modelo W

Para ubicar las pruebas en la misma posición, se integra en el modelo una segunda V dedicada a las pruebas. Ambas V unidas, dan la W del modelo W.

Los 4 ajustes de interacción necesarios entre pruebas y los cambios en la implementación son aclarados en el lado derecho del modelo W.

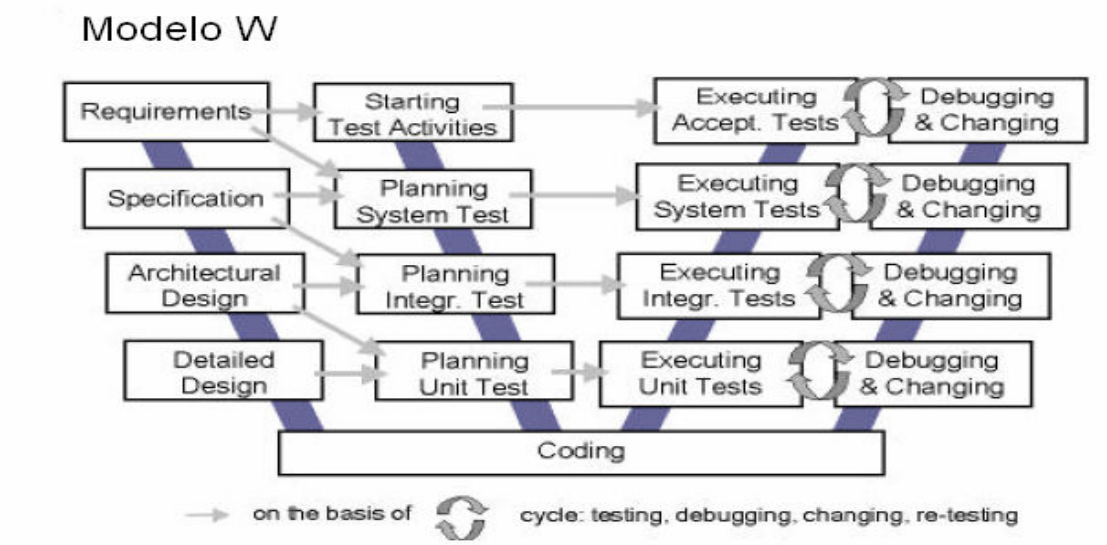

Fig. 2 Modelo de Prueba W [Spillner, 2000]

Este lado contiene no solo las actividades de pruebas destructivas como en el modelo V, sino que también las actividades de cambio constructivas que son llevadas a cabo como resultado de la detección de fallas y defectos.

Ventajas

- En el modelo W, la importancia de las pruebas y el orden de las actividades individuales para las pruebas está clara. Paralelo al proceso de desarrollo, en un sentido ajustado, se realiza otro proceso –el proceso de pruebas –. Este no es primeramente iniciado después que se ha completado el desarrollo.
- Desde el mismo inicio los probadores y los desarrolladores son confiados con las mismas tareas y son vistos como compañeros de trabajo con iguales derechos. Durante la fase de prueba, el desarrollador es responsable de la eliminación de los defectos y la corrección de la implementación. La temprana

colaboración y la cooperación entre los dos grupos, en la práctica, puede con frecuencia evitar conflictos.

- El modelo W se acerca a la práctica cuando los gastos de las pruebas están dados en el 40 % o más. El modelo claramente enfatiza el hecho de que las pruebas es solo construcción, ejecución y evaluación de los casos de pruebas.

Desventajas

- No aclaran los gastos necesarios para los recursos que necesitan ser asignados a las actividades individuales.
- En el modelo W aparece que las diferentes actividades tienen un requerimiento igual para los recursos (tiempo, personal, etc.)

En el modelo W, las pruebas son consistentemente mostradas como un proceso separado que tiene una cerrada interconexión con las actividades de desarrollo. El modelo de pruebas clarifica aun más la prioridad de las áreas y la dependencia entre las actividades de desarrollo y las de pruebas. [Spillner, 2000]

1.5.3 Disciplina de Pruebas de RUP

Se encarga de evaluar la calidad del producto en desarrollo, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Esta disciplina brinda soporte a las otras disciplinas. Sus objetivos son:

- Encontrar y documentar defectos en la calidad del software.
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

Las actividades de este flujo comienzan pronto en el proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de las prueba en el proyecto, así como las estrategias y recursos con que se dotará a esta tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida sirva para ir refinando el producto a desarrollar.
El objetivo de las pruebas en RUP es producir con éxito el producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios y al cuerpo de ventas.
- Migrar el software existente o convertir bases de datos.

Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito del flujo es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Su ejecución inicia en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, en la elaboración del manual de usuario y tutoriales.

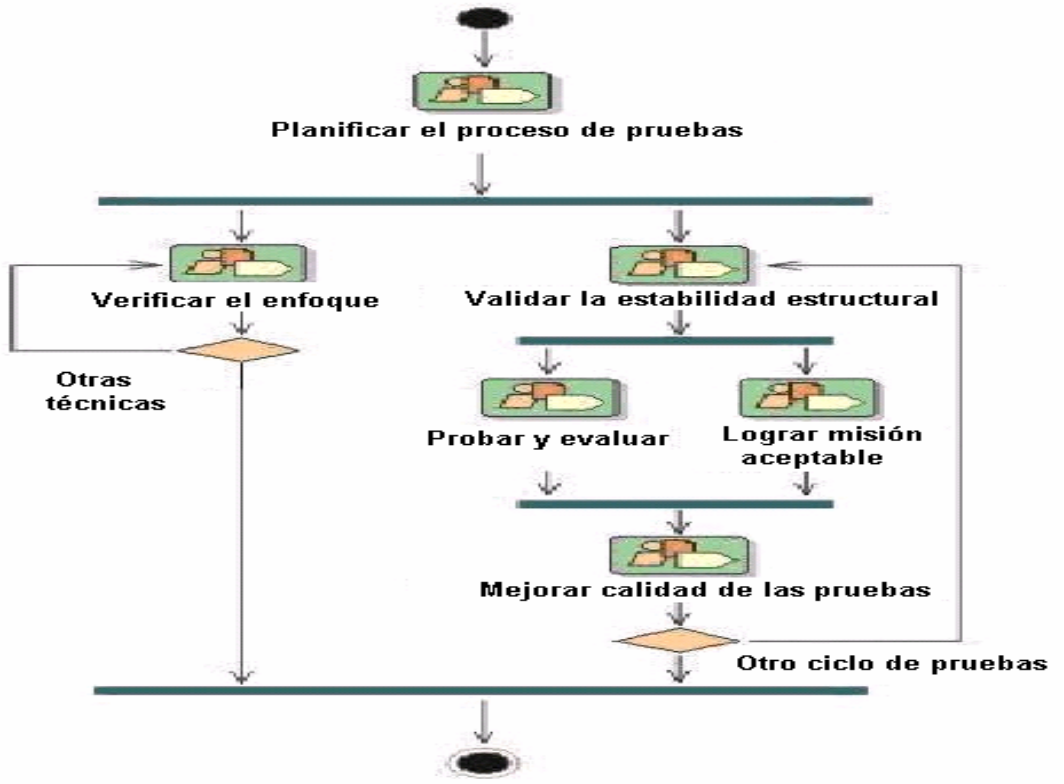

Fig.3 Modelo de Prueba de RUP [RUP, 2003].

1.6 Automatización de las Pruebas

Es una práctica muy beneficiosa realizar las pruebas de manera automática, mediante herramientas de software. [Rice, 2002] enumera y explica los diez retos más importantes en la automatización del proceso de pruebas. De acuerdo con este autor, éstos son los siguientes:

1. Falta de un proceso básico de pruebas y de conocimiento de qué es lo que se debe probar.
    a) Falta de soporte o comprensión por parte de los jefes, debido otra vez a la escasa importancia que habitualmente se le da a la fase de pruebas.
    b) Organización inadecuada del equipo de pruebas.

2. Formación inadecuada en el uso de la herramienta.
3. Falta de uso de las herramientas de prueba que ya se poseen, bien por su dificultad de uso, por falta de tiempo para aprender a manejarla, por falta de soporte técnico, obsolescencia, etc.

4. Falta de herramientas, debida fundamentalmente a su elevado precio o a que las existentes no se ajusten al propósito o entorno para el que se necesitan. La primera razón parece deberse a la a importancia que habitualmente se le da a la fase de pruebas, y eso que el costo de corregir un error puede, en muchos casos, superar al de la licencia de uso. Sería conveniente evaluar el coste de corrección de defectos del software entregado y compararlo con el de la licencia de la herramienta de pruebas.
5. Falta de compatibilidad e interoperabilidad entre herramientas.
6. La herramienta no cubre todos los tipos de prueba que se desean (corrección, fiabilidad, seguridad, rendimiento, etc.). Obviamente, a la hora de elegir la herramienta, deberían tenerse priorizados los tipos de pruebas, y entonces hacer la elección de la herramienta basados en esto. A veces también es necesario utilizar no una, sino varias herramientas de prueba, así como tener en cuenta que es imposible automatizar el 100% de las pruebas.
7. Falta de proceso de gestión de la configuración. Igual que las diferentes versiones del código fuente, las pruebas, especialmente las de regresión, deben someterse a un control de versiones. Recuérdese que el proceso de Gestión de la Configuración es uno de los procesos de soporte del estándar ISO/IEC 12207 (ISO/IEC, 1995),que debería utilizarse en la ejecución de los procesos principales, y muy especialmente en los de Desarrollo y Mantenimiento.
8. Adquisición de una herramienta inadecuada.

1.6.1 Tipos de Herramientas. Ejemplos.

Existen varias clasificaciones para las herramientas que existen para hacer más fácil el trabajo de las pruebas a software, entre ellas:

- Herramientas para el diseño y el desarrollo de las pruebas
- Herramientas para la ejecución y evaluación de pruebas
- Herramientas de asistencia a las pruebas de software
- Herramientas para la documentación de las pruebas

1.6.1.1 Herramientas para el diseño y el desarrollo de las pruebas

El diseño de las pruebas es el proceso de detallar la metodología completa especificada en el plan de pruebas, identificando y priorizando los casos de pruebas asociados. El desarrollo de pruebas es el proceso de traducir el diseño en casos de pruebas específicos.

Esta categoría incluye herramientas para preparar los datos y los casos de pruebas que pueden requerir algún formato específico. Como en la planificación estas herramientas no aportan gran ayuda en el proceso de diseño de las pruebas, el cual es mayormente un proceso mental. Sin embargo, las herramientas de la categoría de ejecución y evaluación ayudan con el desarrollo de las pruebas y son las más útiles para implementar casos de pruebas que han sido apropiadamente planeados y diseñados.

Los tipos de herramientas necesarias para el diseño y desarrollo de pruebas son:

- extractores de datos
- generadores de datos de pruebas
- herramientas de diseño de pruebas basadas en los requerimientos
- capture/ playback
- análisis de cobertura

La herramienta de generación de datos de pruebas automatiza la generación de datos de prueba, basándose en un formato definido por el usuario, por ejemplo, automáticamente generando permutaciones de una entrada específica para una operación especificada por el usuario.

La herramienta de diseño de pruebas basada en los requerimientos, se basa en la suposición de que el 80% de los costos por errores se deben a requerimientos fallidos. Es usada para diseñar casos de pruebas, con el objetivo de asegurar que todos los sistemas implementados cumplen con los requerimientos formalmente especificados Los extractores de datos construyen datos de pruebas a partir de bases de datos o conjuntos de pruebas existentes.

1.6.1.2 Herramientas para la ejecución y evaluación de pruebas

La ejecución y evaluación de pruebas es el proceso que consiste en ejecutar casos de pruebas y evaluar los resultados obtenidos. Esto incluye seleccionar los casos de pruebas que serán ejecutados, establecer o recrear el ambiente de ejecución, ejecutar las pruebas seleccionadas, registrar las actividades de ejecución, analizar fallas potenciales del producto y medir la efectividad del esfuerzo.

Las herramientas en la categoría de evaluación y ejecución ayudan en el proceso de ejecutar los casos de pruebas y evaluar los resultados, analizando dinámicamente el software a ser probado y automatizando la actividad de comprobar los resultados de

las pruebas, la cual usualmente es una actividad tediosa, propensa a fallas y consume mucho tiempo.

Los tipos de herramientas necesarias para la ejecución y evaluación de las pruebas son:

Capture/Playback	Automatizan la ejecución de pruebas, evitando que las pruebas tengan que re-ejecutarse repetidamente en forma manual. Capturan operaciones del usuario automáticamente registrando las entradas de las pruebas (scripts de captura). Estas pruebas capturadas, incluyendo la salida que ha sido validada por el probador, forman una línea base que permitirá su reutilización en pruebas subsecuentes, luego de realizarse cambios en el código. De esta manera la herramienta puede automáticamente retroceder y ejecutar las pruebas capturadas previamente cuando sea necesario y valida los resultados obtenidos comparándolos con los resultados previos capturados en la línea base.
Análisis de Cobertura	Son un medio de descubrir si el software está siendo minuciosamente probado. La herramienta informa cuales partes del producto en evaluación, han sido en realidad ejecutadas por las pruebas actuales.
Pruebas de Memoria	En general, las herramientas en esta categoría poseen la capacidad para detectar: - problemas de memoria; - el uso de memoria fuera del espacio de memoria del programa; - memoria asignada pero no liberada; - lectura y utilización de memoria no inicializada
Administradores de casos de pruebas	Complementan la tarea realizada por una herramienta de capture/ replay, orientada a construir y automatizar los pruebas, asistiendo en la organización y administración de las pruebas obtenidas. Los mejores administradores de casos de pruebas:

	- proveen una interfaz de usuario para administrar pruebas - organizan pruebas para facilitar su uso y mantenimiento - comienzan y administran sesiones de ejecución de pruebas que ejecutan pruebas seleccionadas por el usuario - proveen una integración completa con las herramientas de cobertura y captura/replay - proveen reporte y documentación automatizada de las pruebas.
Simuladores y herramientas de pruebas de rendimiento	Los simuladores toman el lugar del software o hardware que interactúa con el software a ser probado. En general, las herramientas de rendimiento ayudan a determinar cuales son las capacidades de rendimiento del software y del sistema, monitoreando características de tiempos de los componentes del sistema o de sistemas enteros. Además, existen herramientas para automatizar pruebas de carga en sistemas multiusuarios cliente/ servidor y para analizar el rendimiento de los mismos.
Debuggers	Usualmente asisten en forma directa las pruebas, aún cuando su principal objetivo es localizar errores resultantes de las pruebas previamente realizadas. Los debuggers pueden ser usados para desempeñar varias funciones de pruebas de bajo nivel y son las únicas herramientas de ejecución de pruebas que algunas organizaciones tienen.
Analizadores de red	Estas herramientas tienen la capacidad de analizar el tráfico de una red para identificar áreas de problemas y condiciones actuales de la misma. Estos analizadores usualmente permiten simular las actividades de múltiples terminales.
Gestor de ejecuciones	Es una clasificación general de las herramientas de pruebas que automatizan la configuración, la ejecución y la documentación de las pruebas.

Comparadores	Comparan los resultados de las pruebas del programa con los resultados de pruebas anteriores reportando las diferencias entre ellos. Los comparadores son esenciales en pruebas de regresión donde se comparan los resultados de ejecutar las versiones viejas y nuevas.
Analizadores y reductores de datos	Convierten los datos a un formato más sencillo de interpretar por el usuario. En ciertos casos pueden desarrollar análisis estadísticos sobre los datos.
Rastreadores de defectos y cambios	Mantienen una traza con información de los errores y generan informes sobre los mismos. Son usualmente una parte de los sistemas de gestión de la configuración.

Tabla 2 Herramientas para pruebas de software [RUP, 2003]

1.6.1.3 Herramientas de asistencia a las pruebas de software

Esta categoría incluye herramientas que, pese a que no son la base del proceso de pruebas prestan un soporte global a esta actividad.

Los tipos de herramientas principales para asistir las pruebas son:

Gestión de problemas

Estas herramientas son usadas para registrar, rastrear y generalmente asistir en la administración de defectos y mejoras a lo largo del ciclo de vida de los productos de software.

Gestión de configuración

Es la herramienta clave para administrar, controlar y coordinar cambios a documentos, y cualquier otra cosa que sea de interés. Estas herramientas asisten al control de versiones y al proceso de administración de construcción.

Gestores de proyecto

Ayudan a los líderes a planificar el desarrollo y mantenimiento de un sistema. Estas herramientas documentan las estimaciones, planeamientos, requerimientos de recursos y progreso de todas las actividades del proyecto incluyendo las pruebas.

1.6.1.4 Herramientas para la documentación de las pruebas

Los tipos de herramientas necesarias para la planificación de las pruebas son:

- Plantillas para documentar el plan de prueba.
- Planificadores de pruebas y estimadores de personal.
- Analizadores de complejidad.

1.6.1.5 Adquisición de herramientas

La adquisición de herramientas para las pruebas es un tema más que nada de buen sentido común, pero es difícil y duro de implementar.

Tomar decisiones sobre la adquisición de herramientas debe involucrar un análisis simple de costo/beneficio. Esto generará una estimación, en la cual no sólo deberá considerarse el precio de la licencia o de la compra sino también los costos adicionales, tales como costos de instalación, operación, entrenamiento, mantenimiento, asistencia y el costo general de reorganizar los procesos.

Además, en el momento de adquirir una herramienta será necesario identificar como asistirá ésta al proceso de las pruebas, se debe conocer de antemano como se planificará y diseñarán las pruebas (las herramientas no eliminan la necesidad de pensar, planear y diseñar), determinar si se tiene el apropiado entrenamiento para utilizar la nueva herramienta y si el uso prolongado de la misma será promovido dentro de la organización.

La relación de herramientas de mayor utilidad para los entornos de pruebas es:

Compañía	Herramienta	Características
Compuware	File-AID	Es una herramienta que permite la gestión segura y eficiente de grandes ficheros de datos de producción, en línea y batch, en muy diferentes plataformas y entornos.
	TestPartner	Es una herramienta que automatiza las pruebas funcionales y de regresión. Ha sido especialmente diseñada para complejas aplicaciones basadas en Microsoft, Java y tecnologías Web.

	TrackRecord	Se ajusta a cualquier proceso de desarrollo y pruebas, ofreciendo un sistema de rastreo que ayuda en la identificación y resolución de defectos de software.
Parasoft	JTest, C++ Test y Insure++	Herramientas de Parasoft, que ayudan a prevenir errores por medio de su capacidad de análisis estático personalizable, que permite hacer cumplir automáticamente alrededor de 300 estándares de codificación de la industria, crear y hacer cumplir estándares propios de codificación, o construir estándares destinados a un proyecto o grupo en particular.
	TEST	TEST es una unidad de pruebas automatizada y productos de análisis de código estándar, que trabaja sobre clases escritas en la plataforma Microsoft .NET, sin requerir que los desarrolladores realicen un solo caso de prueba o stub.
Fortify	Software Security Manager	Manager es una herramienta de control de seguridad para la gestión de riesgos derivados del software, está orientada para los equipos de desarrollo y seguridad. Además de proporcionar informes flexibles, esta herramienta aporta una gestión centralizada de reglas, políticas de seguridad y alertas.

Tabla 3 Herramientas de entorno de pruebas [Falcato, 2002].

Otras herramientas automatizadas existentes son:

- Analizador de código: es una especie de depurador externo que examina además de las sentencias, los caminos e hilos del programa.
- Analizador de cobertura: solamente prueba las ramas y caminos de todas las funciones que trabajan, internamente o externamente, con el programa.

- Analizador de memoria: evalúa si las aplicaciones no exceden los límites de memoria en los peores casos, o situaciones criticas.
- Perfomance de carga: en sistemas cliente-servidor, esfuerza al programa para que trabaje con cargas pesadas y en situaciones extremas.
- Analizador de sitios WEB: examina los enlaces y las aplicaciones en los diferentes nodos y en el servidor.
- Reporte de bugs: trabaja en conjunto con analizadores de código y de cobertura, haciendo un reporte de las partes de códigos no examinadas o confusas.
- Reporte de configuración: hace un reporte de los requerimientos del programa en cuestión a hardware y los parámetros mínimos que se deben cumplir para que el software pueda trabajar al máximo.
- Reporte de desempeño: hace un reporte del nivel de desempeño, aplicación por aplicación, del programa en el hardware instalado. [Falcato, 2002]

1.6.2 Ventajas de la Automatización

- Mayor rapidez de ejecución.
- Menos recursos.
- Se evitan pruebas obsoletas.
- Se evitan fallos humanos.

1.6.3 Desventajas de la Automatización

- Muy pocas herramientas.
- Necesitan una “base” a menudo inexistente.
- Un conjunto pobre de pruebas.

1.7 Conclusiones del Capítulo

Las pruebas a software son un aspecto determinante en la calidad de procesos de desarrollo de software y por tanto de los productos que se obtengan de estos. A través de un adecuado proceso de pruebas se puede lograr un software libre de defectos que no cause el descontento de los clientes, lo que permitiría a las empresas ganar confiabilidad y prestigio en este mercado.

Capítulo 2 Diseño del Proceso de Pruebas

Introducción

Para lograr un software de gestión libre de errores y en menor tiempo posible se necesita un adecuado proceso de pruebas de software. El objetivo de este capítulo fue realizar un estudio del proceso de pruebas para software de gestión en la Universidad, para determinar los aspectos positivos y negativos del mismo, y proponer un nuevo proceso de pruebas para software de gestión que cumpla con las características requeridas para asegurar, en el mayor porcentaje posible, la ausencia de defectos en los productos que se exportan, y con ello la calidad y fiabilidad de los mismos.

2.1 Características generales del proceso de pruebas en la Universidad de las Ciencias Informáticas

El procesamiento y estudio de la encuesta aplicada a varias personas implicadas en el asesoramiento de la calidad a un total de 21 proyectos productivos de la universidad, permitió hacer una caracterización del proceso de pruebas a software de gestión en la misma. [Ver Anexo 4]

Este estudio permitió conocer que en la universidad el proceso de prueba se sigue es por el modelo de pruebas que plantea RUP, sobre todo de una forma encaminada a encontrar y exponer debilidades en el producto de software. Los resultados que arrojaron dichas encuestas son los siguientes:

1. En cuanto al proceso de pruebas se obtuvo que:

    - Solo el 52.4% de los proyectos de gestión de la universidad siguen un proceso de pruebas.
    - El 76.2% de los proyecto planifican sus pruebas.
    - El 76.2% de los proyectos encuestados que producen software de gestión se guían por una estrategia de prueba para llevar a cabo las pruebas al software.
    - EL 47.6% de los proyectos encuestados le realizan las pruebas al software que obtienen durante todo el ciclo de vida del mismo, mientras que el otro 52.4% realiza las pruebas al obtener el producto final.

- El 76.2% de los encuestados afirman que dan capacitación al equipo de prueba para llevar a cabo el proceso de prueba exitosamente.
- El 0% de los proyecto de gestión siguen un estándar de calidad para la realización del proceso de pruebas.
- El 0% de los proyecto realizan pruebas al tener una versión del sistema.

2. En cuanto a los métodos de pruebas que se utilizan se tiene que:

    - El 14.3% de los proyectos encuestados utilizan el método de prueba de caja blanca.
    - El 90.5% de los proyectos de gestión encuestados utilizan el método de prueba de caja negra.

3. En cuanto a los niveles de pruebas que se emplean se tiene que:

    - El 14.3 % de los proyectos implementan pruebas en el nivel de Unidad.
    - El 42.9% de los encuestados implementan pruebas en el nivel de Integración.
    - El 33.3% de los proyectos implementan las pruebas de Sistema, posibilitando la integración de todo el sistema.
    - El 9.5% de los proyectos que producen software de gestión realizan pruebas de Regresión, comprobando la posibilidad de no introducir nuevos errores adicionales en los cambios
    - El 71.4% de los encuestados implementan pruebas de Aceptación, verificando que el software esté listo para su entrega.

A continuación se muestra el grafico que refleja la utilización de los niveles de pruebas.

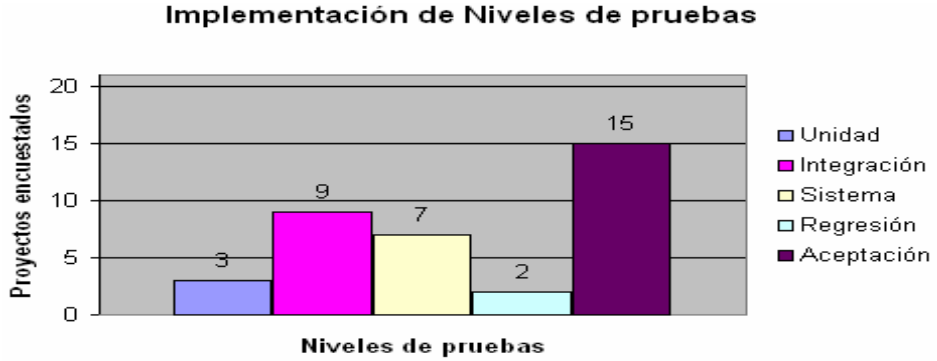

Fig. 4 Niveles de Pruebas.

4. En cuanto al tipo de pruebas que se aplican, se tiene que:

    - Solo el 81% de los proyectos realizan pruebas de Funcionalidad.
    - El 76.2% de los proyectos de gestión realizan pruebas de Seguridad
    - El 42.9% de los proyectos que producen software de gestión llevan a cabo pruebas de Usabilidad.
    - Solo el 28.6% de los encuestados aplican pruebas de Configuración.
    - El 28.6% de ellos realizan pruebas de Instalación.
    - El 23.8% aplican pruebas de Volumen.
    - Solo el 9.5% de los proyectos encuestados realizan pruebas de Rendimiento.
    - El 9.5% de los proyectos de gestión llevan a cabo pruebas de Stress.

5. En cuanto a los artefactos de salida que se exige como documentación en los proceso de pruebas se tiene que:

    - El 66.7% de los proyectos encuestados exigen la entrega de un Plan de Pruebas.
    - El 66.7% exigen como documentación del flujo de trabajo prueba la entrega de la Estrategia de pruebas.
    - El 61.9% exigen la entrega como documentación de la Descripción de Arquitectura de software.
    - Solo el 81% de los encuestados exigen la entrega del documento de Resultados de las pruebas.
    - El 81% de los proyectos de gestión exigen la entrega del Documento de No Conformidades.
    - Solo el 57.1% de los proyectos que producen software de gestión exigen la entrega de un Manual de Usuarios.
    - Solo el 76,2% de los encuestados exigen la Descripción de los Casos de Uso como documentación del flujo de trabajo de prueba.

6. En cuanto a la utilización de herramientas que automaticen las pruebas se tiene que:

- Solo el 19% de los proyecto encuestados utilizan algunas herramientas para la automatización de las pruebas, donde las más utilizadas son las herramientas libres JUnit, JMeter y Unit Test.

Otro estudio realizado fue la encuesta que fueron aplicadas a un total de 4 directivos del área de calidad de la Universidad, con el fin de conocer sus opiniones sobre el estado de las pruebas en la UCI [Ver Anexo 7], las cuales mostraron los siguientes criterios:

1. En cuanto al estado actual del proceso de pruebas en la UCI se tiene que:

- El 75% de los encuestados plantean que en la UCI no se realizan apropiadamente los procesos de pruebas, consideran que el estado de las pruebas en los proyectos de la universidad es insuficiente.
- El 75% del personal encuestado opinan que en la universidad no existe una adecuada estructura organizativa en los proyectos.
- El 75% plantea que existe un profundo desconocimiento de las técnicas de pruebas que se deben aplicar, en dependencia del tipo de software que se este construyendo.
- El 75% de los encuestados plantea que no existen personal capacitado en este tema y disponible para llevar a cabo un proceso de pruebas.
- El 75% del personal encuestado plantea que la estimación de los proyectos es inadecuada, lo que provoca un diseño y ejecución de las pruebas en corto tiempo y con recursos deficitarios.

A continuación se presenta un gráfico que muestra las causas del estado ineficiente de las pruebas en la UCI.

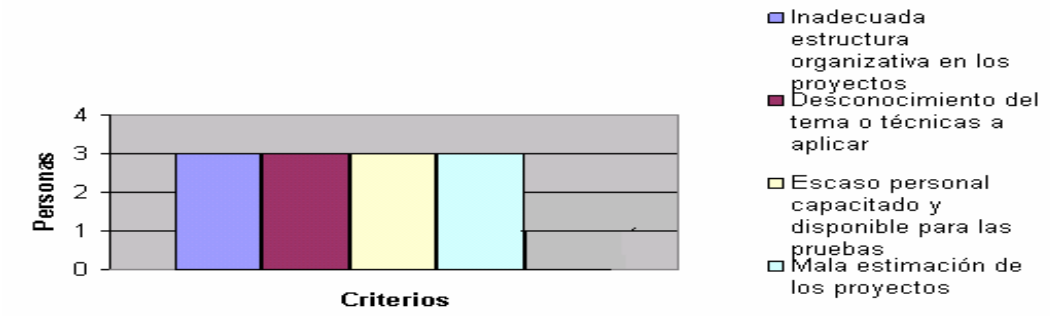

Fig. 5 Estado de las Pruebas en la UCI.

Es una realidad que la universidad no cuenta con el suficiente personal capacitado como para llevar a cabo un proceso de pruebas adecuado en cada proyecto productivo, lo que en ocasiones provoca que el equipo de pruebas que se designa no realice eficientemente su trabajo. Sucede que no se conoce a profundidad el tema de la ingeniería de pruebas, y de todos los aspectos que esta encierra, necesarios para garantizar la calidad de cualquier proyecto de producción de software, y esto está dado porque no se lleva a cabo un vasto proceso de capacitación a trabajadores y estudiantes para el dominio del tema.

Se puede decir al respecto que no se cuenta con un sitio centralizado o repositorio de pruebas que almacene toda la documentación referente al proceso de prueba que se debe desarrollar en cada uno de los proyectos, lo que impide que la documentación que se genere durante el desarrollo de cualquier proyecto pueda ser consultada por cualquier miembro del equipo de calidad y que no exista información registrada referentes a las pruebas del software.

Por otro lado, la organización dentro de los proyectos es mayormente deficiente, por lo general no se estima adecuadamente el tiempo y costo de desarrollo de los proyectos, lo que provoca que a la hora de planificar el tiempo de cada una de las etapas del proceso de producción, la asignación de tiempos y recursos que se le da a la etapa de pruebas no es adecuada, quedando estas para realizar al final del proceso, cuando ya se tiene una versión del producto y en ocasiones ya casi la fecha de entrega del producto está a término.

En la mayoría de los casos sí se planifican las pruebas, pero no se traza una estrategia a seguir para llevarlas a cabo, así como tampoco se sigue ningún estándar que garantice en alguna medida que los procesos de prueba se realicen de forma satisfactoria, de manera que no se siguen adecuadamente pautas que dictan normas de prestigio internacional como CMMI, ISO, etc.

No se aplican todas las pruebas necesarias para garantizar la máxima localización y eliminación de los defectos, así como tampoco se implementan todos los niveles de pruebas necesarios para garantizar este aspecto. Un nivel de prueba muy importante que en muchos casos no se aplica adecuadamente es el de unidad, o sea que prácticamente no se realizan pruebas de caja blanca, al menos de una manera adecuada, que permitan corregir la mayoría de los errores introducidos por los

programadores en el código que generan, donde eliminar dichos errores es menos costoso que en etapas posteriores. En muchos casos, son los propios programadores los que realizan verificaciones a su código, pero no aplican las técnicas requeridas, o sea que estas no son realmente pruebas en sí, y son muchos los errores que pasan inadvertidos por ello.

Por otro lado, en la universidad se utilizan pocas herramientas para la automatización de las pruebas, las cuales facilitan enormemente este proceso, y ganan cada vez mayor aceptación en empresas de software de gran prestigio internacional, como una solución fiable en procesos de pruebas.

Finalmente se puede decir que aunque en la universidad se están dando los primeros paso, aún falta mucho por conocer y avanzar con respecto al tema de los procesos de pruebas a software, en vistas de que este es un aspecto vital para lograr la calidad requerida para comercializar los productos que se obtengan, y lograr de este modo el objetivo primordial, que es ubicar esta empresa en un punto cimero en el mercado mundial de software.

De manera general se llegó a la conclusión de que el estado actual de las pruebas a software en la UCI es insuficiente, lo que arroja como resultado que en muchos proyectos productivos el tiempo y costo empleado sea superior a lo estimado.

2.2 Propuesta de un Proceso de Pruebas de la Universidad de las Ciencias Informáticas

El Proceso de Pruebas para Software de Gestión de la Universidad de las Ciencias Informáticas que se propone se ha estructurado para llevar un correcto control de los elementos que se han probado, y los resultado obtenidos, así como de los elementos a probar. Se realizó sobre la base del proceso de pruebas que propone el modelo RUP, con adaptaciones específicas para la UCI y utilizando también la planificación de los niveles de pruebas que propone el modelo V. A continuación se presentan las etapas de este proceso.

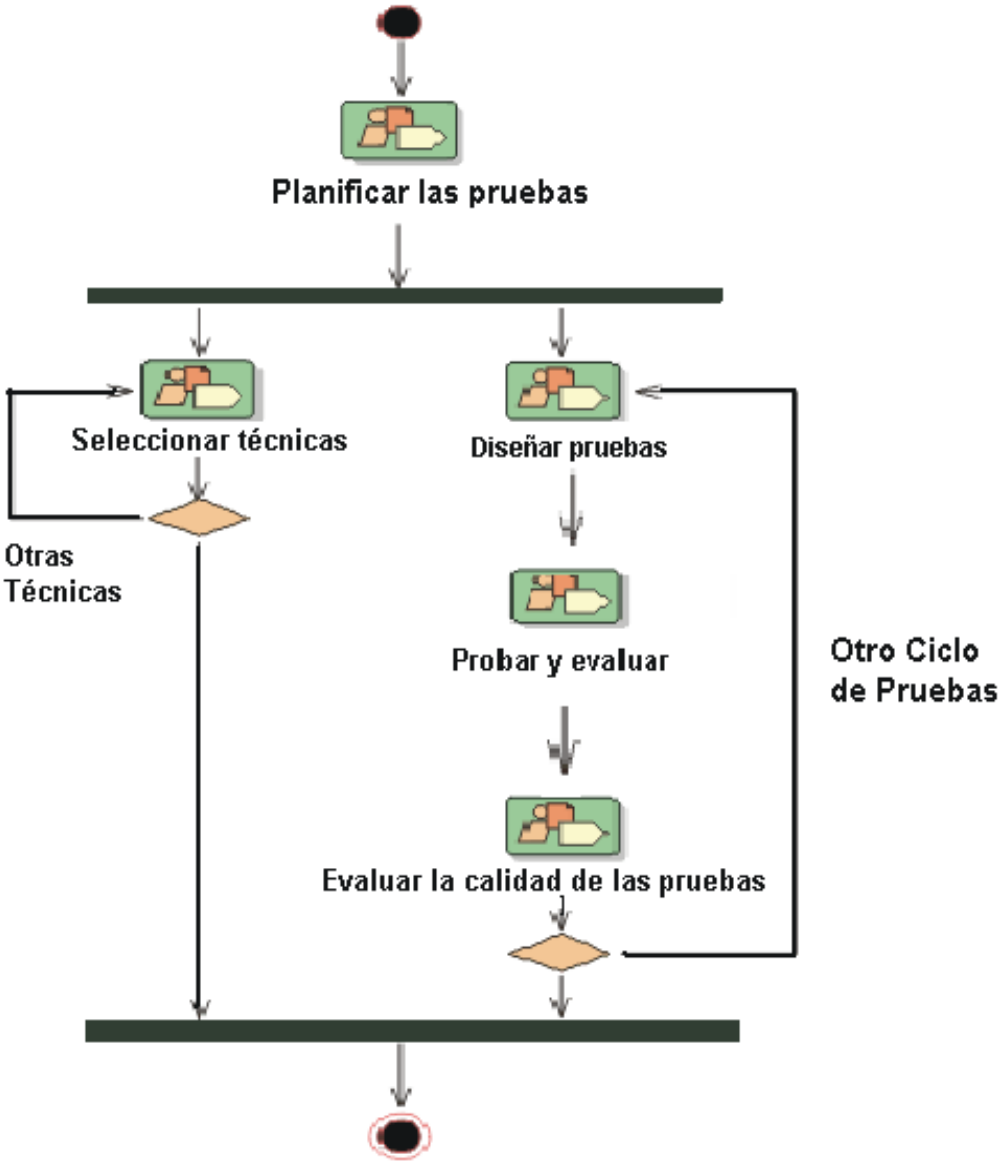

Fig. 6 Flujo de Trabajo del Proceso de Pruebas para Software de Gestión.

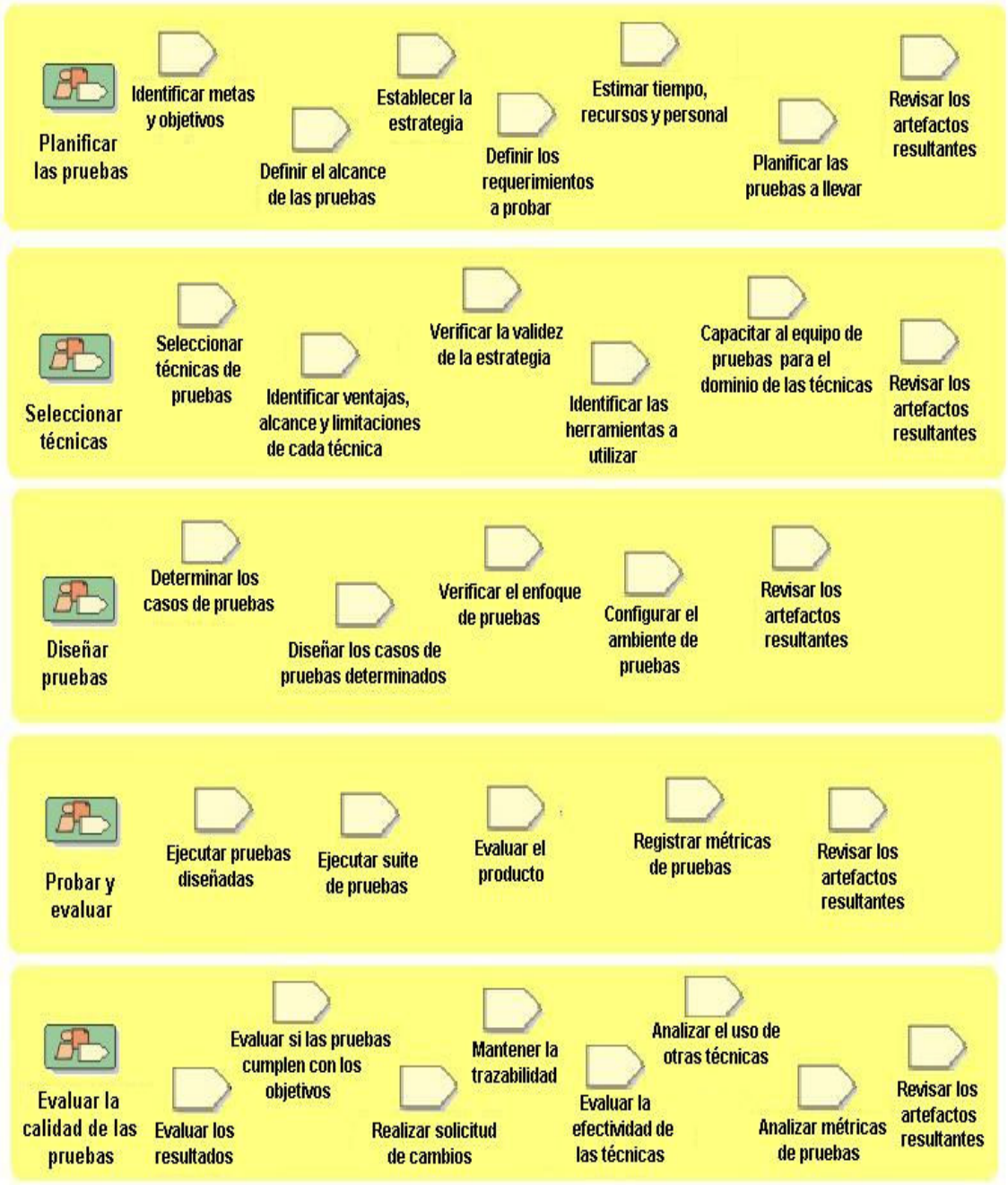

Fig. 7 Proceso de Prueba detallado con sus actividades.

A continuación se detallarán las actividades, artefactos de entrada y salida involucrados en este proceso.

2.2.1 Flujo de Trabajo: Planificar las pruebas.

Su objetivo principal es la planificación exhaustiva del esfuerzo de prueba. Se determinan los objetivos de las pruebas, se identifica el método apropiado de prueba para cada iteración, y se llega a un acuerdo entre todos lo involucrados en el proceso de pruebas con respecto a los recursos, el alcance y cronograma del mismo, quedando estos aspectos definidos en un plan de pruebas. Dicho plan puede sufrir cambios durante la aplicación del proceso, por lo que el mismo constituye meramente una guía para el proceso de pruebas.

Actividades

- Identificar las metas y objetivos de las pruebas para cada iteración.
- Definir el alcance de las pruebas.
- Establecer la estrategia de pruebas a seguir durante el proceso.
- Definir los requerimientos a probar.
- Estimar tiempo, recursos y personal requerido para las pruebas.
- Planificar las pruebas a llevar a cabo durante el proceso.
- Revisar los artefactos resultantes.

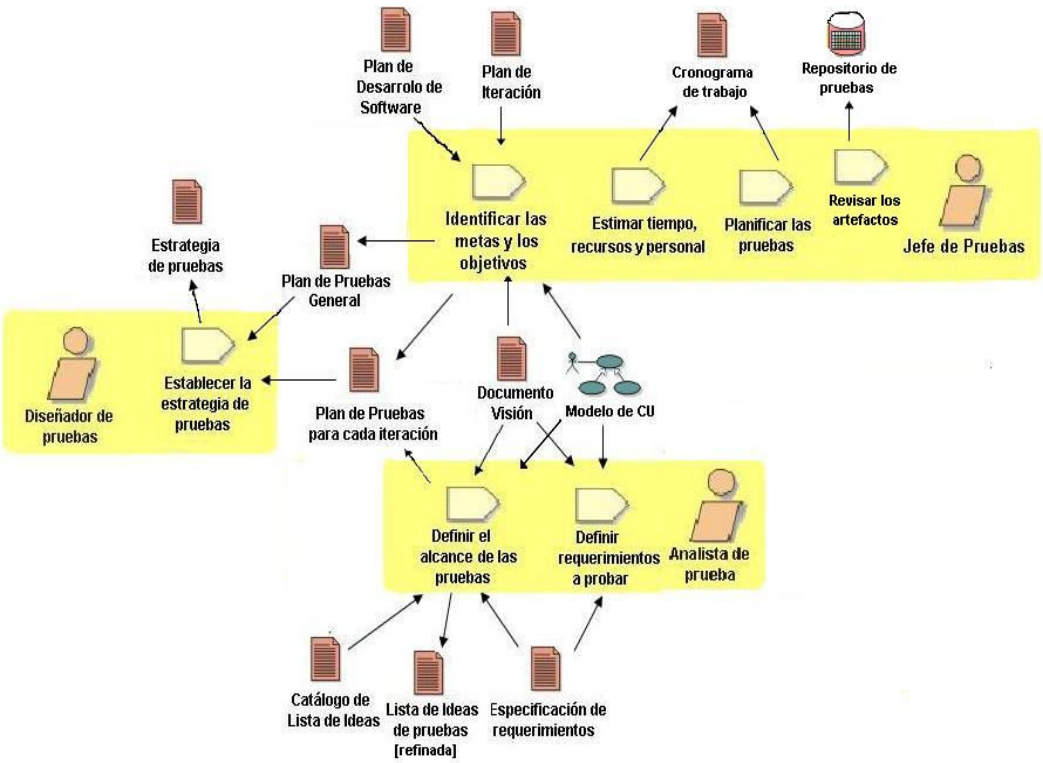

Fig. 8 Flujo de Trabajo: Planificar Pruebas.

Identificar las metas y objetivos de las pruebas para cada iteración.

Esta actividad tiene como misión principal identificar los elementos que serán probados, tanto de software como de hardware, para solicitar la información adecuada para la planificación de los tipos de pruebas que se van a implementar, en dependencia del tipo de software que será probado.

Definir el alcance de las pruebas.

El propósito de esta actividad es definir el alcance y las limitaciones que pueden tener las pruebas seleccionadas, según el software que será probado.

Establecer la estrategia de pruebas a seguir durante el proceso.

Esta actividad tiene como objetivo definir una estrategia de prueba que incluye también todas las técnicas de diseño de los casos de pruebas que van a guiar el proceso de prueba a lo largo del ciclo de vida del software.

Definir los requerimientos a probar.

Después que se obtiene la lista de requerimientos del software, es necesario definir cuales van a ser los requerimientos candidatos o importantes que necesitan ser verificados para continuar con la construcción del software y que, por lo tanto, deben ser probados.

Estimar tiempo, recursos y personal requerido para las pruebas.

El propósito de esta actividad es planificar desde un inicio el tiempo, los recursos que son necesarios y el personal que conformará el equipo de pruebas, para llevar a cabo el proceso de prueba en el software que se está construyendo. Con el avance de las pruebas se debe verificar que el tiempo planificado sea igual al tiempo real que se está empleando.

Planificar las pruebas a llevar a cabo durante el proceso.

El propósito de esta actividad es determinar las pruebas que se van a aplicar durante todo el proceso de prueba. Las pruebas se planifican por fase según el Modelo V. En la fase de inicio se debe ir planificando las pruebas de aceptación, para validar requerimientos. En la fase de Elaboración se debe planificar las pruebas de sistema para verificar el diseño del software. En la fase de Construcción se debe planificar las pruebas de integración, para lograr un diseño detallado, además de las pruebas de unidad, para ir probando todas las estructuras individuales que se van obteniendo

durante la implementación del software. Para la fase de Transición se debe ejecutar y refinar las pruebas de aceptación, para verificar que lo que se concibió al inicio del proyecto sea lo que se obtenga al final de la construcción del software.

Revisar los artefactos resultantes

Esta actividad tiene como propósito revisar los artefactos que se obtienen durante la ejecución de todas las actividades y guardarlos en el repositorio de pruebas.

2.2.2 Flujo de Trabajo: Seleccionar técnicas.

Su función fundamental es determinar las técnicas apropiadas que faciliten el trabajo de las pruebas, de manera que se produzcan resultados precisos, con los recursos disponibles. El objetivo es lograr un entendimiento general de las restricciones y limitaciones de cada técnica al aplicarlas a los proyectos de la universidad.

Actividades

- Seleccionar técnicas de pruebas apropiadas a ejecutar.
- Identificar las ventajas, el alcance y las limitaciones de cada técnica.
- Verificar la validez de la estrategia de pruebas.
- Identificar herramientas a utilizar.
- Capacitar al equipo de pruebas para el dominio total de las técnicas a ejecutar seleccionadas.
- Revisar los artefactos resultantes.

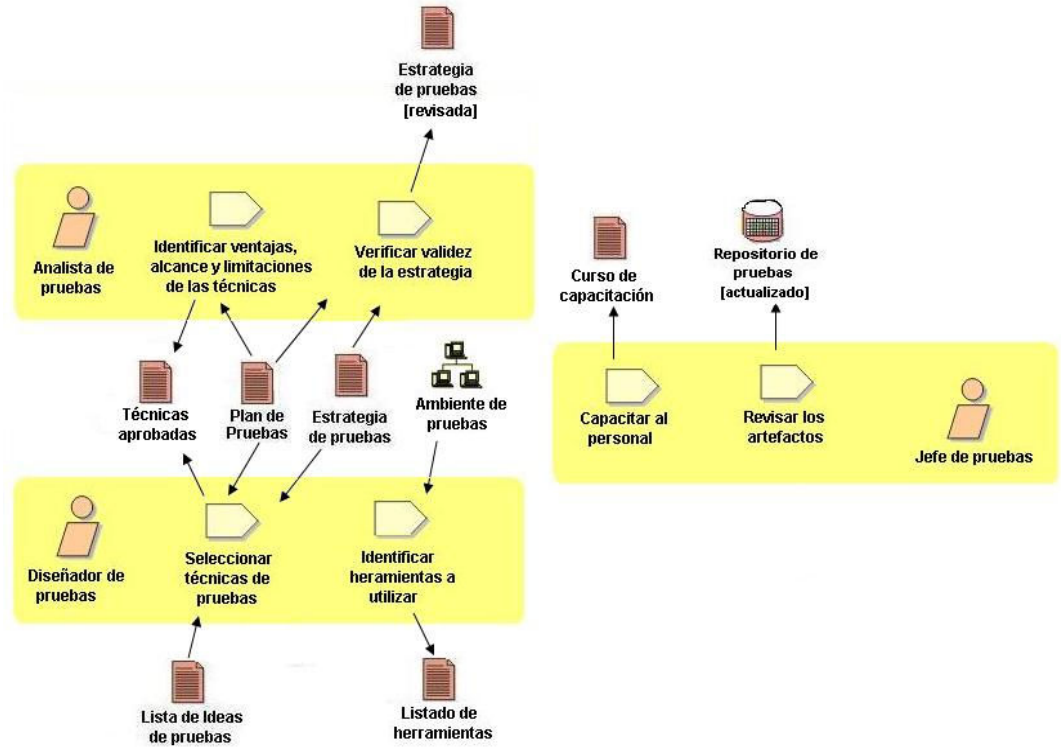

Fig.9 Flujo de Trabajo: Seleccionar Técnicas de Pruebas.

Seleccionar técnicas de pruebas apropiadas a ejecutar

Esta actividad tiene como propósito la selección de todas las técnicas de pruebas necesarias para probar todas las construcciones que se vayan realizando a lo largo del ciclo de vida del software.

Identificar las ventajas, el alcance y las limitaciones de cada técnica

Esta actividad tiene como propósito conocer las ventajas, el alcance y las limitaciones que puedan tener las técnicas seleccionadas, para verificar si pueden ser aplicadas sobre el producto que se va a probar.

Verificar la validez de la estrategia de pruebas

Esta actividad tiene como propósito comprobar que la estrategia de prueba trazada al inicio de la planificación de las pruebas, para el proyecto, proporcionará resultados satisfactorios al llevar a cabo el ciclo de pruebas en el software, o sea que a través de los métodos y técnicas de pruebas seleccionados se podrán encontrar y corregir todos los posibles errores surgidos en el proceso o finalmente en el producto.

Identificar herramientas a utilizar.

Esta actividad tiene como propósito escoger las herramientas de pruebas que van a ser utilizadas durante el proceso de pruebas, en dependencia de las pruebas que no sea factible realizar manualmente y de la selección de técnicas de pruebas y de los tipos de pruebas.

Capacitar al equipo de pruebas para el dominio total de las técnicas a ejecutar seleccionadas.

Esta actividad tiene como propósito enseñar al equipo de pruebas que participará en este proceso toda la información referente a las técnicas y herramientas seleccionadas en actividades anteriores para llevar a cabo un mejor proceso de prueba.

Revisar los artefactos resultantes

Esta actividad tiene como propósito revisar los artefactos que se obtienen durante la ejecución de todas las actividades y guardarlos en el repositorio de pruebas.

2.2.3 Flujo de Trabajo: Diseñar las pruebas.

El propósito general de este flujo de trabajo es definir y diseñar los casos de pruebas que guiarán el proceso de pruebas planificado, documentando la secuencia de pasos para ejecutar cada caso de prueba, los requisitos de datos para realizar las pruebas, los resultados esperados y los métodos para registrar los resultados.

Actividades

- Determinar casos de pruebas a desarrollar durante el proceso planificado.
- Diseñar los casos de pruebas determinados.
- Verificar el enfoque de prueba.
- Configurar el ambiente de prueba.
- Revisar los artefactos resultantes.

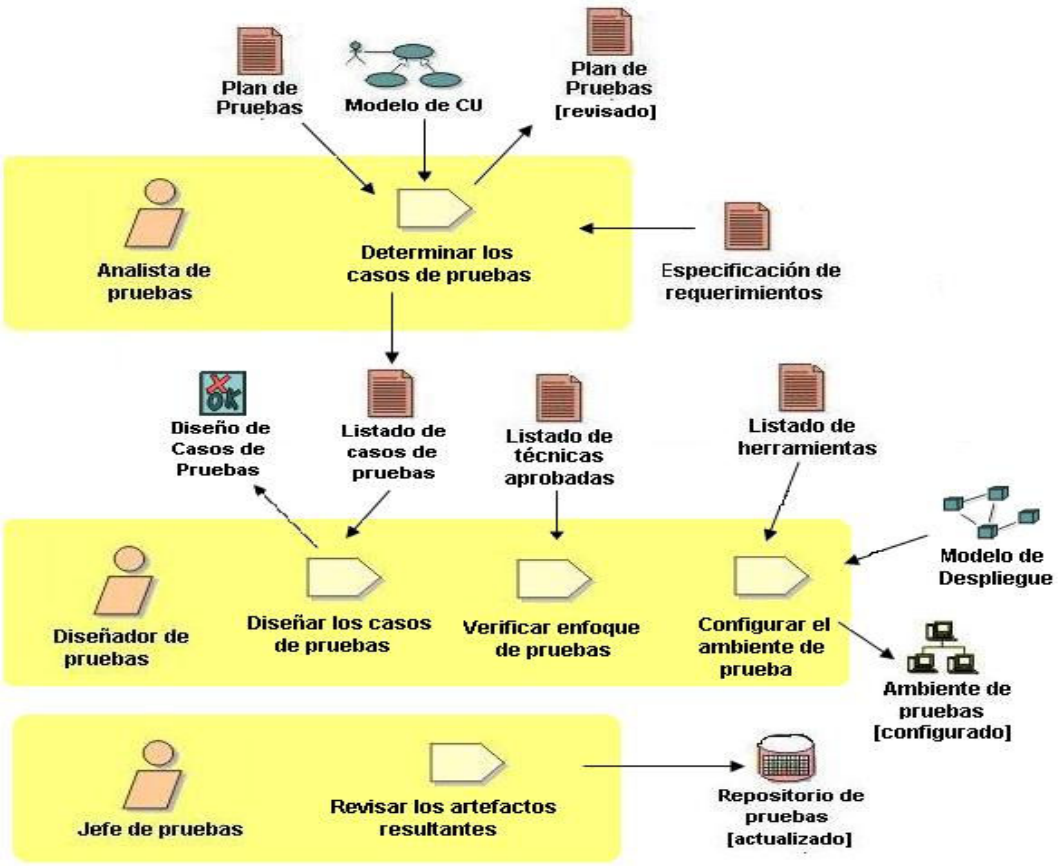

Fig. 10 Flujo de Trabajo: Diseñar Pruebas.

Determinar casos de pruebas a desarrollar durante el proceso planificado.

Teniendo en cuenta los casos de uso, los requerimientos a probar previamente seleccionados y las técnicas de pruebas seleccionadas, se determinan los casos de pruebas a ejecutar.

Diseñar los casos de pruebas determinados.

El propósito de esta actividad es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Los casos de pruebas seleccionados quedan descritos en la plantilla destinada para ello: Plantilla de diseño de Casos de Pruebas.

Verificar el enfoque de prueba.

Se comprueba que las técnicas a utilizar facilitarán el trabajo de las pruebas planificadas, sobre la base de los casos de pruebas, el plan de pruebas y las

herramientas previamente seleccionadas, verificando además que sean apropiadas, teniendo en cuenta los recursos disponibles.

Configurar el ambiente de prueba.

Su objetivo es definir los elementos de hardware y software que servirán de soporte al proceso de pruebas, teniendo en cuenta las pruebas planificadas y las técnicas a implementar.

Revisar los artefactos resultantes

Esta actividad tiene como propósito revisar los artefactos que se obtienen durante la ejecución de todas las actividades y guardarlos en el repositorio de pruebas.

2.2.4 Flujo de Trabajo: Evaluar y Probar

En este flujo se ejecutan las pruebas diseñadas con el fin de hacer una evaluación profunda de los requerimientos a probar definidos en el Plan de Pruebas, y determinar si estos han sido cumplidos.

Actividades

- Ejecutar las pruebas diseñadas
- Ejecutar suite de pruebas.
- Evaluar el producto.
- Registrar métricas de pruebas.
- Revisar los artefactos resultantes.

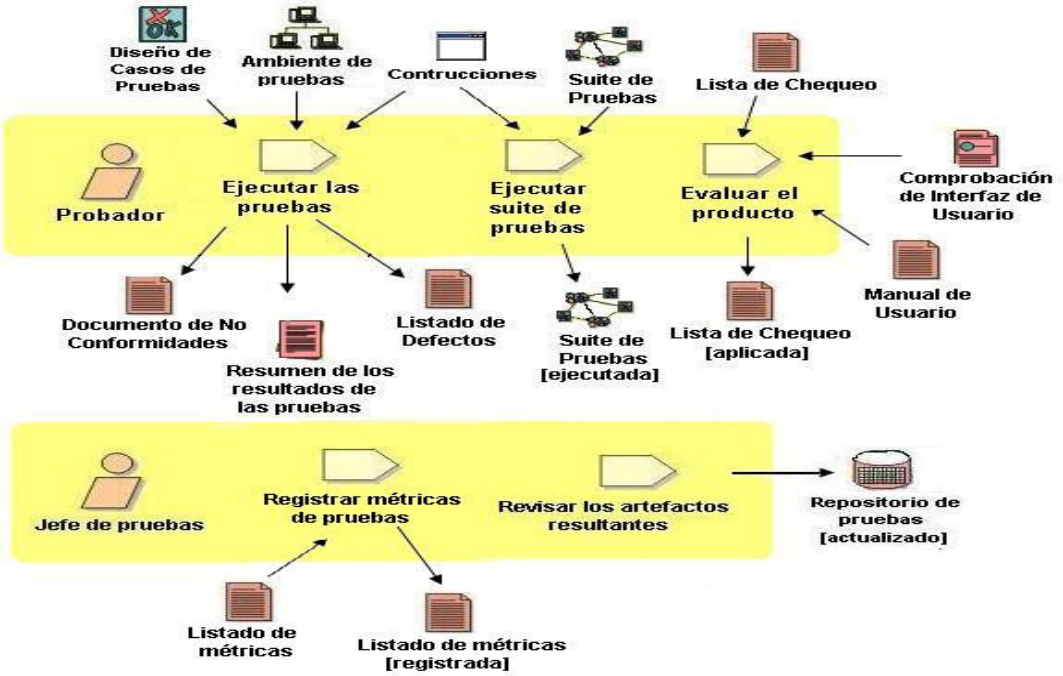

Fig. 11 Flujo de trabajo: Probar y Evaluar

Ejecutar las pruebas diseñadas

Se ejecutan los casos de pruebas previamente diseñados en un ambiente de pruebas disponible, ya sea manualmente o a través de una herramienta, según esté especificado en el diseño del caso de prueba.

Ejecutar suite de pruebas

Tiene como propósito fundamental ejecutar una apropiada colección de pruebas requeridas para evaluar la calidad del producto a través de las herramientas de automatización.

Evaluar el producto

Esta actividad tiene como propósito evaluar el producto resultante a partir de la utilización de listas de chequeos.

Registrar métricas de pruebas

Tiene como propósito aplicar métricas de pruebas para evaluar la completitud y calidad de las mismas.

Revisar los artefactos resultantes

Esta actividad tiene como propósito revisar los artefactos que se obtienen durante la ejecución de todas las actividades y guardarlos en el repositorio de pruebas.

2.2.5 Flujo de Trabajo: Evaluar la calidad de las pruebas

Este flujo de trabajo está orientado a mostrar a los involucrados en el proceso de pruebas si los resultados del proceso cumplen con los objetivo y las metas trazadas en el Plan de Pruebas, de manera que finalmente se hayan cumplido los requerimientos del cliente y el producto está totalmente listo para ser desplegado, de lo contrario, se define un nuevo ciclo de pruebas a realizar.

Actividades

- Evaluar los resultados de las pruebas
- Evaluar si las pruebas realizadas cumplen los objetivos y metas descritos en el Plan de Pruebas.
- Realizar solicitud de cambio.
- Mantener trazabilidad de las pruebas.
- Evaluar la efectividad de las técnicas de pruebas utilizadas.
- Analizar la probabilidad de utilizar otras técnicas no aplicadas.
- Analizar métricas de pruebas.
- Revisar los artefactos resultantes.

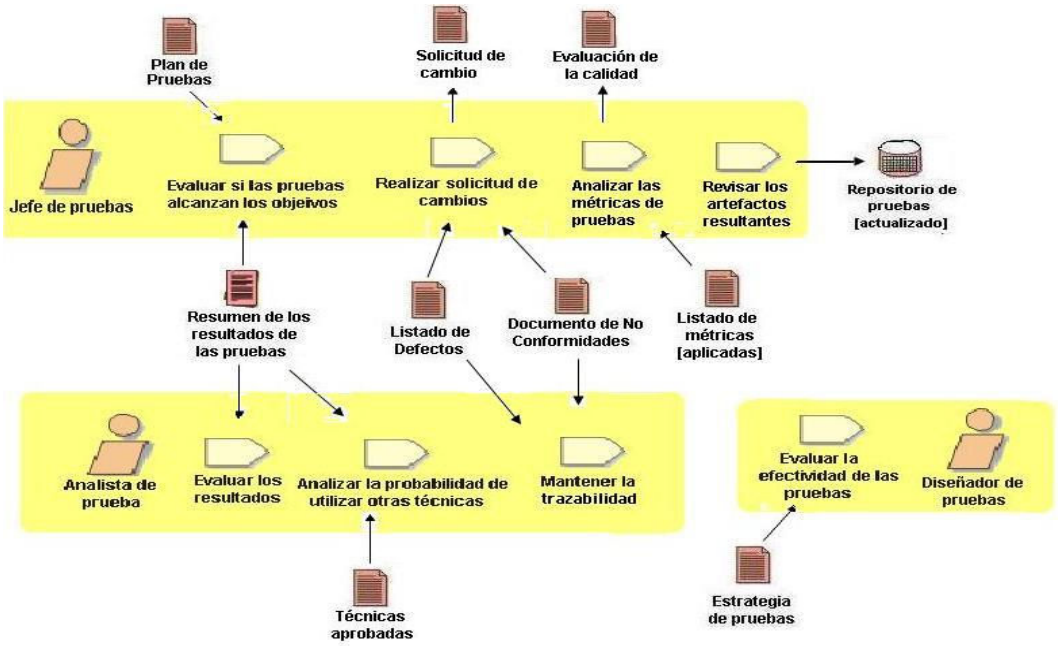

Fig. 12 Flujo de Trabajo: Evaluar la calidad de las pruebas

Evaluar los resultados de las pruebas

Se examinan todos los errores y fallos encontrados durante la ejecución de las pruebas, y se analiza el estado de los mismos, evaluando si ya han sido solucionados, o hay que darle seguimiento.

Evaluar si las pruebas realizadas alcanzan los objetivos y metas descritos en el Plan de Pruebas.

Se determina si cada una de las pruebas ha alcanzado plenamente las metas y objetivos trazados inicialmente en el Plan de Pruebas, y se determina si se culmina el ciclo de pruebas y/o si es necesario hacer una solicitud de cambio.

Realizar solicitud de cambio.

Sobre la base de las metas y los objetivos no alcanzados en las pruebas realizadas, se solicita al Comité de Cambios, el tratamiento a errores no solucionados.

Mantener trazabilidad de las pruebas.

Se le da seguimiento a los errores que se van encontrando durante el proceso de pruebas, hasta garantizar la total corrección de los mismos.

Evaluar la efectividad de las técnicas de pruebas utilizadas

Se evalúa en qué medida las técnicas de pruebas seleccionadas han cumplido con las metas trazadas en el Plan de Pruebas

Analizar la probabilidad de utilizar otras técnicas no aplicadas.

Se determina la posibilidad de utilizar otras técnicas en un nuevo ciclo de pruebas, en caso de que las técnicas aplicadas no hayan cumplido con las metas trazadas en el Plan de Pruebas.

Analizar métricas de pruebas

Tiene como propósito analizar los resultados de la aplicación de las métricas de pruebas y determinar si las pruebas planificadas cumplen con los objetivos y metas trazados.

Revisar los artefactos resultantes

Esta actividad tiene como propósito revisar los artefactos que se obtienen durante la ejecución de todas las actividades y guardarlos en el repositorio de pruebas.

2.3 Capacitación del personal

Para la capacitación del personal involucrado en los proceso de pruebas se decidió realizar dos curso de capacitación que permitió a los involucrados aumentar sus conocimientos a un nivel mas alto.

2.3.1 Curso de Capacitación sobre Proceso de Prueba

Con la amplia actividad productiva que tienen las distintas facultades de la UCI, ha existido desde su inicio un incremento de la necesidad de existencia de personal calificados en temas como la calidad de software y dentro de ella que tengan conocimientos en cuanto a la aplicación de procesos de pruebas a software.

Existe en todas las facultades el perfil de Calidad de Software y los que pertenecen a dicho perfil, se dedican en todo momento a la preparación sistemática en temas como las pruebas a software para garantizar la entrega de los productos que se construyen, libres de errores, en tiempo y con un gran nivel de aceptación por el cliente.

Por este motivo y dándole cumplimiento al objetivo de Capacitación al Personal vinculado a los procesos de pruebas a software de gestión, se presenta en forma de curso optativo, la propuesta de la asignatura de “Introducción a las Pruebas de Software” y dirigido a estudiantes que cursan el primer año.

UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS
DEPARTAMENTO DE INGENIERÍA Y GESTIÓN DE SOFTWARE
DIRECCIÓN DE CALIDAD DE SOFTWARE
PROGRAMA ANALITICO CURSOS OPTATIVOS

Datos Generales			
Disciplina:	**Ingeniería y gestión de software**		
Curso Optativo:	**Introducción a las Pruebas de Software**		
Perfil:	**Segundo Perfil de Calidad de Software**		
Año:	1er Año	**Semestre:**	**2do**
Duración Total:	**20 Horas**		

Tema	**C**	**CP**	**CTP**	**S**	**T**	**L**	**OTRA**	**Evaluación**	**Total**
Tema 1	4		8		8				**20**
Totales	**4**		**8**		**8**				**20**

Objetivos Generales

- Definir los elementos y conceptos de pruebas de software.
- Interpretar el proceso de pruebas de software.
- Aplicar Casos de pruebas y listas de chequeos de interfaz y de atributos de calidad.
- Registrar resultados de pruebas, los errores, métricas y no conformidades.
- Elaborar el reporte de no conformidad.

Sistema de Valores

La responsabilidad, la sensibilidad hacia la necesidad del trabajo con los procesos de pruebas en los proyectos, el compromiso, el sentido de pertenencia que debe tener hacia la universidad y mucha habilidad para adaptarse a los cambios que se puedan producir.

Descripción de los Temas

Tema 1 Introducción a las Pruebas.

Objetivos:

- Definir el concepto de pruebas y los elementos fundamentales de las pruebas.
- Caracterizar las técnicas de prueba que existen.
- Caracterizar las herramientas de pruebas que existen.

Sistema de conocimientos:

Introducción a los elementos fundamentales de las pruebas de software: niveles, métodos, tipos, artefactos y modelos. Las distintas técnicas de pruebas que existen y estudio de las herramientas que facilitan la automatización de las pruebas.

Tema 2: Aplicación de los casos de pruebas

Objetivos:

- Caracterizar el Flujo de Trabajo de Prueba.
- Organizar un equipo de trabajo para la realización de pruebas.
- Aplicar los casos de prueba diseñados a partir de un caso de uso correctamente definido y asociado a un requerimiento.
- Elaborar el reporte de no conformidad.

Sistema de conocimientos:

Introducción a las características generales del Flujo de trabajo de Prueba. Organización de un equipo de trabajo de prueba. Definición de roles. Requerimientos, asociación con un CU. Evaluación de los CU. Aplicar casos de pruebas. Lista de chequeos y su aplicación. Reporte de no conformidad.

Evaluación del Tema:

La evaluación será a través de las actividades sistemáticas. Además los estudiantes deben desarrollar un caso de estudio, donde aplicarán casos de pruebas y realizarán todo el procedimiento de prueba. Se evaluará la productividad.

Indicaciones de organización de la asignatura

Los estudiantes deben tener conocimientos sobre el proceso de desarrollo de software y alguna metodología de desarrollo.

Sistema de evaluación de la asignatura

Está basado en evaluaciones sistemáticas tanto orales como escritas en los talleres, en las clases teórica prácticas y un trabajo final.

Los estudiantes deben desarrollar un caso de estudio, donde identificaran a partir de un conjunto de pruebas, las que son necesarias en dependencia del software que será probado.

La asignatura tendrá como trabajo final la realización de pruebas de aceptación, o de liberación de un software.

Bibliografía

1. Tesis “Propuesta de un Proceso de Prueba para proyecto de Software de Gestión”, Telma Rodríguez Alfonso y Adisleydis Olano Montero, 2007.
2. “El Proceso Unificado de Desarrollo de Software”. Iver Jacobson, Grady Booch y James Rumbwgh. Addison-Wesley. 2000.
3. “Ingeniería de Software: Un enfoque práctico”. Roger Pressman. Mc Graw-Hill/interamericana de España. 2002.
4. Ayuda Ampliada de RUP. 2003.

A continuación se presenta el curso optativo “Procesos de Pruebas” dirigido a estudiantes de tercer año.

UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS
DEPARTAMENTO DE INGENIERÍA Y GESTIÓN DE SOFTWARE
DIRECCIÓN DE CALIDAD DE SOFTWARE
PROGRAMA ANALITICO CURSOS OPTATIVOS

Datos Generales			
Disciplina:	**Ingeniería y gestión de software**		
Curso Optativo:	**Proceso de Pruebas de Software**		
Perfil:	**Segundo Perfil de Calidad de Software**		
Año:	3er Año	**Semestre:**	**2do**
Duración Total:	**20 Horas**		

Tema	C	CP	CTP	S	T	L	OTRA	Evaluación	Total
Tema 1	4		8		8				**20**
Totales	**4**		**8**		**8**				**20**

Objetivos Generales

- Planificar pruebas.
- Diseñar casos de pruebas de software
- Evaluar producto.
- Identificar características de calidad.
- Aplicar métricas para la evaluación de software.
- Ejecutar pruebas, registrar resultados de pruebas.
- Analizar resultados de pruebas ejecutadas y de las métricas aplicadas.

Sistema de Valores

La responsabilidad, la sensibilidad hacia la necesidad del trabajo con los procesos de pruebas en los proyectos, el compromiso, el sentido de pertenencia que debe tener hacia la universidad y mucha habilidad para adaptarse a los cambios que se puedan producir.

Descripción de los Temas

Tema 1 Diseñar Casos de pruebas.

Objetivos:

- Diseñar de los casos de pruebas.
- Seleccionar de técnicas de pruebas según el software que se va probar.
- Aplicar las plantillas para la confección de la documentación del proceso.

Sistema de conocimientos:

Definición de casos de pruebas. Diseño de casos de pruebas. Plantillas de la documentación de pruebas. Métricas para la evaluación de las pruebas. Registrar resultados de la aplicación de las métricas.

Tema 2. Planificación de las pruebas

Objetivos:

- Definir una buena estrategia de prueba.
- Planificar de las pruebas según el software que se va a probar.

Sistema de conocimientos:

Plan de pruebas. Estrategia de prueba. Planificación de las pruebas y asignación de recursos.

Tema 3: Evaluando un software

Objetivos:

- Identificar características de calidad.
- Aplicar métricas para la evaluación de software.

Sistema de conocimientos:

Introducción a la Calidad de Software. Características de calidad. Metodología de evaluación. Métricas para evaluar software. Proceso de consenso. Reunión de Expertos.

Evaluación del Tema:

La evaluación será a través de las actividades sistemáticas. Además los estudiantes deben desarrollar un caso de estudio, donde aplicaran métricas. Se evaluará la productividad con el cuaderno de trabajo.

Indicaciones de organización de la asignatura

Los estudiantes deben tener conocimientos sobre las pruebas de software, conocer la disciplina de prueba de RUP y conocer algunos tipos de técnicas de pruebas. Deben haber acreditado el curso optativo Introducción a las Pruebas.

Sistema de evaluación de la asignatura

Está basado en evaluaciones sistemáticas tanto orales como escritas en los talleres y en las clases teórica prácticas y un trabajo final.

Los estudiantes deben desarrollar un caso de estudio, donde sean capaces de desarrollar los distintos roles que implica un proceso de prueba.

Los estudiantes deben ser capaces de planificar, diseñar, seleccionar las técnicas adecuadas, aplicar las pruebas de un software y evaluar los resultados.

La asignatura tendrá como trabajo final el desarrollo de un proceso de pruebas a partir de un caso de estudio que se le entregará donde incluirá desde la planificación hasta análisis de los resultados.

Bibliografía

1. Tesis “Propuesta de un Proceso de Prueba para proyecto de Software de Gestión”, Telma Rodríguez Alfonso y Adisleydis Olano Montero, 2007.
2. “El Proceso Unificado de Desarrollo de Software”. Iver Jacobson, Grady Booch y James Rumbwgh. Addison-Wesley. 2000.
3. “Ingeniería de Software: Un enfoque práctico”. Roger Pressman. Mc Graw-Hill/interamericana de España. 2002.
4. Ayuda Ampliada de RUP. 2003.

Estos dos cursos están apoyados por la propuesta de un Proceso de Prueba para Software de Gestión para la Universidad de las Ciencias Informáticas, aplicable al proceso productivo de la universidad, descrito en este capítulo.

2.4 Conclusiones del Capítulo

Hoy en día se calcula que el proceso de pruebas representa más de la mitad del coste de un programa, ya que requiere un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico cuando estos no involucran vidas humanas, puesto que en este último caso el costo suele superar el 80% siendo este proceso un poco mas costoso que el propio desarrollo y diseño de los distintos programas que conforman el sistema. Un proceso de pruebas requiere mucho más que tiempo y dinero, necesita una verdadera metodología la cual exige herramientas y conocimientos destinados a dicha tarea.

Capítulo 3 Evaluación del Proceso de Prueba

Introducción

Para aceptar y validar el proceso de prueba que se propuso en el capítulo anterior, se conformó un Panel de Expertos que emitió su criterio. Este panel estuvo compuesto por especialistas en el tema de Calidad de la Universidad de las Ciencias Informáticas y de la empresa Softel. Este proceso de validación se llevó a cabo por el Método Delphi, que es una técnica de investigación social que tiene como objeto la obtención de una opinión grupal fidedigna a partir de un grupo de expertos.

3.1 El Método Delphi

Su nombre proviene de la inspiración en el antiguo oráculo de Delphos, fue ideado originalmente a comienzos de los años 50 en el seno del Centro de Investigación estadounidense RAND Corporation por Olaf Helmer y Theodore J. Gordon, como un instrumento para realizar predicciones sobre un caso de catástrofe nuclear. Desde entonces, ha sido utilizado frecuentemente como sistema para obtener información sobre el futuro. Delphi consiste en la selección de un grupo de expertos a los que se les pregunta su opinión sobre cuestiones referidas a acontecimientos del futuro. Las estimaciones de los expertos se realizan en sucesivas rondas, anónimas, con el objeto de tratar de conseguir consenso, pero con la máxima autonomía por parte de los participantes. La encuesta se lleva a cabo de una manera anónima, actualmente es habitual realizarla haciendo uso del correo electrónico o mediante cuestionarios Web establecidos al efecto.

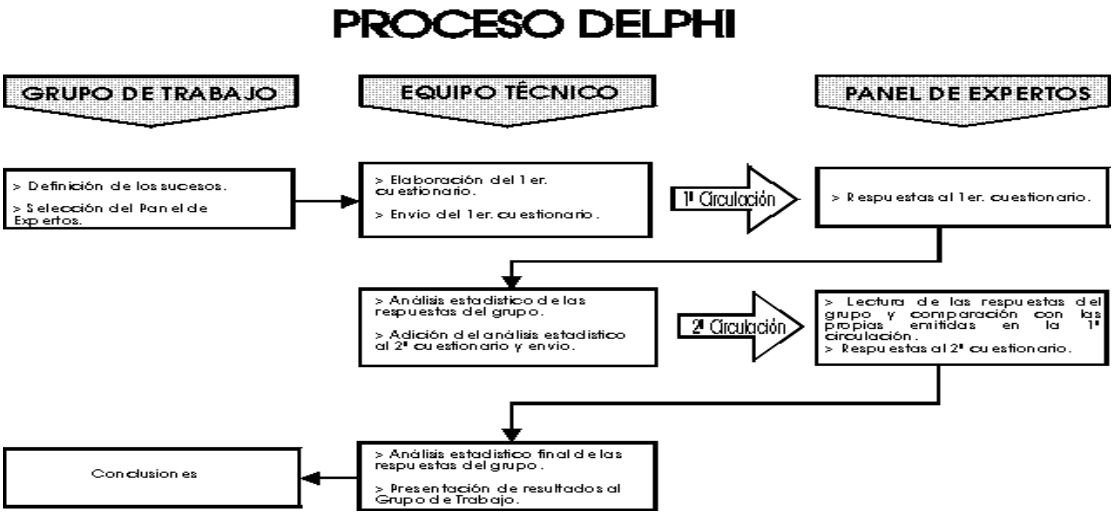

Fig. 13 Proceso Delphi [Linstone, 1975].

3.2 Proceso de selección de expertos

Un experto es una persona o grupo de personas con un gran conocimiento sobre el tema tratado, que puede emitir un criterio concluyente de cualquier problema y emitir valoraciones importantes con un alto nivel de conocimiento. En este proceso de selección del panel de experto se tuvieron en cuenta los siguientes pasos:

3.2.1 Cantidad de expertos que conforman el panel

Para escoger el número de expertos que integraron el panel se tuvo en cuenta los conocimientos que tienen acerca del problema que se plantea y la posible solución que podían emitir. Estos conocimientos están basados en las siguientes áreas de trabajo:

- Proceso Unificado del Software (RUP).
- Proceso de Desarrollo del Software.
- Proceso de Pruebas para Software.
- Calidad del Software.
- Normas y estándares de calidad.
- Métricas de Pruebas.

3.2.2 Listado de expertos

El listado de expertos se realizó con las probabilidades de que tuvieran conocimientos sobre el tema de la calidad y pruebas a software, que fueran personas que estuvieran integradas de forma directa a la producción de software en la UCI, que estuvieran vinculados al área de la Calidad de Software y tuvieran experiencias en los temas vinculados a los procesos de pruebas.

Todas las personas seleccionadas para integrar el Panel de Expertos cuentan con los conocimientos necesarios para emitir una valoración y están vinculados de forma directa a la producción de software en la UCI. Se escogieron un total de 6 expertos, de ellos 2 son especialistas en el departamento central de calidad de la Universidad de las Ciencias Informáticas, 1 es asesor de la calidad en las facultades en donde radica, 1 es directivo de la empresa Softel y 2 son profesoras de ingeniería de software. [Ver Anexo 8].

3.2.3 Consentimiento del experto en su participación

Una vez confeccionado el panel de expertos se invitó a cada uno de ellos de forma personal para que participaran en el proceso de validación y aceptación de la propuesta. Se le detalló de forma clara y precisa los objetivos de este proceso, así como la realización del cuestionario. Con este paso se termina el proceso de selección del personal que conformará el Panel de expertos.

3.3 Elaboración del cuestionario

Para la presentación del cuestionario se tuvo en cuenta realizar preguntas con un enfoque investigativo y que se centraran principalmente en los principios básicos que debe cumplir la propuesta presentada, además de permitir que las repuestas fueran un poco abiertas, y en todos los casos con posibilidad de emitir su criterio personal en cuanto a la propuesta que se presentaba para el proceso productivo de la UCI. [Ver Anexo 9].

En este proceso los expertos que conforman el panel recibieron de forma personal la documentación de la propuesta del proceso de prueba para software de gestión y el cuestionario y se les pidió que completaran esta tarea en un mínimo de tiempo para analizar las respuestas y permitirle que hicieran preguntas en caso que hubiesen surgido dudas durante el estudio de la propuesta.

3.4 Resultados del proceso de validación

El procesamiento y análisis de las respuestas de los cuestionarios se realizó a través de un software estadístico llamado ***SPSS 13.0***, el cual permitió la recopilación de toda la información y ofrecer un resultado final para la validación de la propuesta.

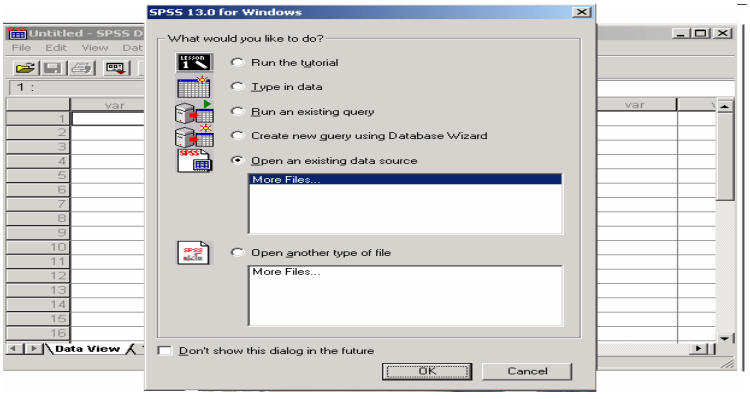

Fig. 14 Software Estadístico SPSS 13.0

La Prueba No Paramétricas o Coeficiente de Kendall

Esta prueba, determina la relación (acuerdo, concordancia) de que las filas (ordenaciones). Es decir, que decimos que no existe concordancia cuando los valores R_j que veremos en la Tabla 6, son bastante parecidos numéricamente. Ahora, el bastante o poco parecido lo veremos más adelante.

Para la Prueba de W de Kendall se colocan los datos en una tabla de doble entraba o doble clasificación, es decir, k filas y N columnas. Ver Tabla 4.

	Criterios						
Experto 1	5	5	5	5	5	5	5
Experto 2	4	4	5	5	4	5	4
Experto 3	2	5	5	5	5	5	4
Experto 4	4	5	5	3	5	5	4
Experto 5	3	5	4	4	5	4	4
Experto 6	5	5	5	4	4	4	5

Tabla 4 Valores emitidos por los expertos.

La Hipótesis:

H_o: Concordancia Nula. No existe concordancia entre los expertos.

H_1: Hay concordancia entre los expertos.

La Prueba:

La prueba de W de Kendall, como planteamos anteriormente observa la variabilidad de los R_j. Así pues, cuando la concordancia entre los expertos es perfecta, la variabilidad entre los R_j es máxima. Y cuando la variabilidad es mínima la concordancia es nula. Con lo planteado anteriormente planteamos el estadístico S como:

$$S = \sum_{i=1}^{N} R_i - \frac{k(N+1)}{2}$$

Desarrollamos ahora el coeficiente de concordancia W,

$$W = \frac{12\sum_{i=1}^{N} {R_i}^2}{k^2 N(N^2-1)} - \frac{3(N+1)}{N-1}$$

El cual toma valores entre 0 cuando la concordancia es nula y 1 cuando es perfecta.

Para hacer la prueba, se toma el estadístico Ω que está distribuido χ^2 con df=N-1 grados de libertad.

$$\Omega = \frac{12}{kN(N+1)} \sum_{j=1}^{N} R_j^2 - 3k(N+1)$$

Statistical Product and Service Solutions (SPSS)

Después de seleccionada las opción ***Analizar -> Pruebas no paramétricas -> K muestras relacionadas…*** nos mostrará en el Visor de Resultados las siguientes Tablas.

La tabla 5 contiene la media por columnas. $\overline{R_j} = \frac{R_j}{k}$.

	Rango promedio
P11	2,30
P22	4,80
P33	4,90
P44	4,00
P55	4,80
P66	4,90
P77	2,30

Tabla 5 Medias por Columnas

La tabla 6 contiene el estadístico Ω, los grados de libertad gl (df) y la significación asintótica. Este último número es el que nos vamos a detener para contrastar la hipótesis H_0.

N	5
W de Kendall[a]	,459
Chi-cuadrado	13,778
gl	6
Sig. asintót.	**,032**

Tabla 6 Estadísticos de contrastes

Puesto que la significación asintótica es 0.032 que es menor que 0.05 (Nivel de significación más usado), podemos rechazar la hipótesis de concordancia nula.

Por lo anterior planteado podemos decir que podemos rechazar la hipótesis de concordancia nula de que no existe concordancia entre los criterios de los expertos y concluimos que existe acuerdo significativo entre los 6 expertos encuestados.

Otras de las opiniones que emitieron los expertos son:

- Es necesario definir un proceso de pruebas al inicio del proyecto teniendo en cuenta que la medición de la calidad del producto, al final del proceso, es muy costosa y es necesario asegurar la calidad en el proceso de producción para lograr un producto exitoso, revisado y libre de errores.

- Es necesario la aplicación de la propuesta a los proyectos productivos de la UCI está dado porque la misma toma como base la metodología de desarrollo de software RUP, que es la más utilizada en los procesos de producción de la universidad, y se imparte en asignaturas como Ingeniería de Software, la forma en que se plantea la propuesta es bastante similar a las prácticas de RUP, lo que dio al traste con un proceso de prueba que debe ser capaz de controlar la calidad en los productos que se desarrollan en la UCI.

- Al aplicar exitosamente la propuesta en los proyectos productivos de la UCI, se facilitaría la planificación desde un inicio de las pruebas que se le van a aplicar al software que será probado, posibilitando el conocimiento previo de la metodología que se está empleando y la publicación de los primeros resultados. Además, aumentaría la disponibilidad de personal capacitado para llevar a cabo un buen proceso de pruebas.

- Se considera importante que el proceso de pruebas propuesto es capaz de realizar las pruebas al producto desde el mismo instante en que se inicia el proyecto, proponiendo un conjunto de actividades que permiten la detección y corrección de errores tempranamente.

- La propuesta cumple con los objetivos de un proceso de pruebas, puesto que define y maneja bien los artefactos que se obtienen en cada actividad y permite la planificación de las pruebas que se van a aplicar en dependencia del software que se va a probar desde el inicio del proyecto.

- Se califica de excelente la idea de introducir cursos de capacitación para el personal que trabajará en las pruebas, porque es importante disponer de recursos humanos calificados para realizar esta tarea de probar y corregir software.

- Como respuesta a la importancia de contar con un proceso de pruebas para la UCI, se opinó que es necesario para el control de la calidad de los productos que se liberan, permitiendo obtener software libres de errores, que cumplan las expectativas de los clientes.

- Los expertos concuerdan que como propuesta inicial a implementar es adecuada en la situación actual que tiene la producción en la UCI, y en que se le debe dar seguimiento hasta lograr institucionalizar el proceso propuesto.

Conclusiones

Una vez concluida la investigación se cumplieron los objetivos planteados, obteniéndose los siguientes resultados:

- Se caracterizó la producción de software en la UCI determinando las causas fundamentales del estado ineficiente del proceso de pruebas en la universidad.
- Se definió un Proceso de Pruebas para Proyectos de Software de Gestión de la UCI, logrando así disminuir considerablemente el tiempo y costo que se suele emplear en esta etapa.
- Se diseñaron dos cursos optativos para la capacitación del personal, en aras de contar con personas preparadas para realizar las pruebas a software en los proyectos productivos de la UCI.

Recomendaciones

En aras de lograr una adecuada puesta en práctica y la obtención de buenos resultados del proceso de pruebas en un proyecto de gestión de la UCI se recomienda:

- Aplicar el proceso a los proyectos de software de gestión e ir refinando a medida que se obtengan resultados tangibles.
- Crear una conciencia en los miembros de los equipos de desarrollo de la necesidad de llevar a cabo un adecuado proceso de pruebas dentro de sus proyectos.
- Capacitar tanto estudiantes como trabajadores, con el fin de contar con el personal requerido para implementar exitosos procesos de pruebas en los proyectos productivos de la universidad.
- Realizar un estudio más profundo de las herramientas de automatización de las pruebas.
- Introducir plantillas que ayuden a registrar y documentar todos los pasos relacionado con las pruebas que se están aplicando, como los diseños y resultados de las pruebas.
- Realizar el plan calendario correspondiente a cada uno de los cursos que se diseñaron.

Referencia Bibliográfica

BEIZER, B. Software testing techniques. Segunda Ed. Van Nostrand Reinhold Company, 1990.

DAVIS, A. Principles of Software Development. Mc Graw, 1995.

FALCATO, M. V. C. Prueba de software España: Editorial Prensa Técnica S. L., [Consultado el: 29 de marzo del 2007 Disponible en: http://html.rincondelvago.com/prueba-de-software.html.

GESTIÓN, D. D. I. D. S. Y. Flujo de trabajo Prueba [Conferencia Digital]. La Habana: [Consultado el: 7 de abril de 2007]. Disponible en: http://teleformacion.uci.cu/file.php/43/Conferencia_5/Conf_FT_Prueba_06-07.pdf.

GURÚ. El Proceso de la Prueba de Software. Lenguaje de Definición de Proceso. Revista Software Gurú, 2005, vol. 01, nº 06, [Consultado el: 8 de febrero del 2007].

HETZEL, H. The Complete Guide to Software Testing. Segunda Ed. Wellesley, MA: QED Information Science, 1988.

IEEE. Standard Glossary of Software Engineering Terminology. IEEE Software Engineering Standards Collection, 1990, nº ISSN 610.12-190.

KANER, C. Lessons learned in software testing. Jonh Wiley, 2002.

KANER, C.; FALK, J., et al. Testing Computer Software. Segunda Ed. Van Nostrang Reinold, 1993.

KIT, E. Software testing in the real world, improving the process. Addison Wesley, 1995.

---. Software Testing in the world Addison-Wesley, 1994.

LINSTONE H., T., M. The Delphi Method. Techniques and Applications. Addison-Wesley, 1975.

MAÑAS, J. A. Prueba de Programas España: [Consultado el: 28 de marzo de 2007]. Disponible en: http://www.lab.dit.upm.es/%7Elprg/material/apuntes/pruebas/testing.htm.

---. Pruebas de Programas [Consultado el: 8 de febrero del 2007 Disponible en: http://www.upv.es/protel/usr/jotrofer/pascal/testing.htm.

MORENO, G. A. Pruebas Granada: [Consultado el: 28 de marzo de 2007]. Disponible en: http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/.

MYERS, B. A. A new model for handling input. ACM Transactions on Information Systems, 1979.

---. A new model for input. ACM Transactions on Information Systems, 1990.

PARETO, D. D. Diagrama de Paretto. El Diagrama de Pareto es una gráfica en donde se organizan diversas clasificaciones de datos por orden descendente, de izquierda a derecha por medio de barras sencillas después de haber reunido los datos para calificar las causas. De modo que se pueda asignar un orden de prioridades. p.

PÉREZ, H. ¿Qué es el testeo? España: [Consultado el: 12 de febrero de 2007]. Disponible en: http://www.astest.com.ar/spip/spip.php?article10.

PÉREZ, Y. F.; DELGADO, R. L. C., et al. Proceso de prueba de aceptación para un software de gestión Cuba: Sin publicar, de 6 de febrero del 2007].

PRESSMAN, R. Ingeniería de Software, Un enfoque practico. Quinta ed. España: Mc Graw-Hill, 2002.

---. Ingeniería de Software. Un enfoque Práctico. España: McGraw- Hill, 2000.

---. Ingeniería de Software. Un enfoque Práctico. España: McGraw- Hill, 1998.

QUESADA, J. A. L. Pruebas de Software [Artículo Digital]. España: [Consultado el: 5 de diciembre de 2006]. Disponible en: http://dis.um.es/~lopezquesada.

RICE, R. W. "Surviving the top 10 challenges of software test automation". Cross Talk: The Journal of Defense Software Engineering, 2002. 26-29 p.

RUP, R. S. C. Ayuda Ampliada de RUP. Estados Unidos: 2003.

SPILLNER, A. The W-MODEL – Strengthening the Bond Between Development and Test Germany: University of Applied Sciences Bremen, [Consultado el: 28 de marzo de 2007].

TERUEL, A. El Plan de Prueba Última actualización: 20 de enero del 2001. [Consultado el: 8 de febrero del 2007 Disponible en: http://www.ldc.usb.ue/~teruel/ci4713/clases2001/planPrueba.html.

---. Requerimiento de Prueba Última actualización: 11 de enero del 2001. [Consultado el: 8 de febrero del 2007 Disponible en: http://www.ldc.usb.ue/~teruel/ci4713/clases2001/testSpecs.html.

TERUEL., P. A. Las Pruebas de Verificación de Requerimientos [Página Web]. Venezuela: Universidad Simón Bolívar de Venezuela, Última actualización: 4 de enero de 2001. [Consultado el: 15 de diciembre de 2006]. (Ingeniería de Software III). Disponible en: http://www.ldc.usb.ue/~teruel/.

Bibliografía

1. ASTIGARRAGA, E. El Método Delphi España: Universidad de Deusto. San Sebastián [Consultado el: 8 de mayo de 2007].

2. BEIZER, B. Software testing techniques. Segunda ed. Van Nostrand Reinhold Company, 1990.

3. COLLADO, M. Prueba del Software. Técnicas y estrategias de pruebas del software. [Consultado el: 8 de febrero del 2007 Disponible en: http:/lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt#284,1,Pruebasdesoftware.

4. CORTÉS, O. H. G. Aplicación práctica del diseño de pruebas de software a nivel de programación Bogotá, Colombia: Willy. Net, [Consultado el: 21 de enero de 2007]. Disponible en: http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf.

5. D`ONOFRIO, D. L. Prueba de Software [Consultado el: 8 de febrero del 2007 Disponible en: http://www.elguille.info/Clipper/probando.htm.

6. DAVIS, A. Principles of Software Development. Mc Graw, 1995.

7. EMILIO. Diseño de la interfaz de usuario: Pruebas del software España: El rincón del vago, [Consultado el: 23 de enero de 2007]. Disponible en: http://pdf.rincondelvago.com/diseno-de-la-interfaz-de-usuario_pruebas-del-software.html.

8. FALCATO, M. V. C. Prueba de software España: Editorial Prensa Técnica S. L., [Consultado el: 29 de marzo del 2007 Disponible en: http://html.rincondelvago.com/prueba-de-software.html.

9. FERRER, J. Programas para la aplicación de métodos de expertos. Ingeniería Industrial, vol. IX No 3 p. 253-256. La Habana, 1998.

10. FEWSTER, M. y GRAHAM, D. Software test automation. Addison Wesley, 1999.

11. GESTIÓN, D. D. I. D. S. Y. Flujo de trabajo Prueba [Conferencia Digital]. La Habana: [Consultado el: 7 de abril de 2007]. Disponible en: http://teleformacion.uci.cu/file.php/43/Conferencia_5/Conf_FT_Prueba_06-07.pdf.

12. GONZALEZ, C. Un plan de Prueba Exitoso [Consultado el: 8 de febrero del 2007 Disponible en: http://www.americaxxi.cl/modules.php?name=News&file=article&sid=20.

13. GURÚ. El Proceso de la Prueba de Software. Lenguaje de Definición de Proceso. Revista Software Gurú, 2005, vol. 01, nº 06, [Consultado el: 8 de febrero del 2007].

14. HERNÁNDEZ, M. A. C. "PRUEBAS DE SOFTWARE" [Pagina Web]. El Salvador: [Consultado el: 10 de enero de 2007]. Disponible en: http://www.monografias.com/trabajos20/pruebas-de-software/pruebas-de-software2.shtml

15. HETZEL, H. The Complete Guide to Software Testing. Second ed. Wellesley, MA: QED Information Science, 1988.

16. IEEE. Standard Glossary of Software Engineering Terminology. IEEE Software Engineering Standards Collection, 1990, nº ISSN 610.12-190.

17. INFORMÁTICAS, S. D. T. Metodología de Trabajo Misiones, República de Argentina: Universidad Nacional de Misiones, [Consultado el: 6 de mayo de 2007]. Disponible en: http://www2.unam.edu.ar/subprograma/metod_fase4.htm.

18. ISO/IEC. International Standard. Software Life Cycle Processes. International Standard Organization/International Electronic Commite, 1995, nº

19. JACABOSON, I.; BOOCH, G., et al. El proceso Unificado del Software. Addison Wesley, 2000.

20. KANER, C. Lessons learned in software testing. Jonh Wiley, 2002.

21. KANER, C.; FALK, J., et al. Testing Computer Software. Segunda ed. Van Nostrang Reinold, 1993.

22. KIT, E. Software testing in the real world, improving the process. Addison Wesley, 1995.

23. ---. Software Testing in the world Addison-Wesley, 1994.

24. LINK, J. Unit testing in Java. Morgan Kaufmann, 2003.

25. LINSTONE H., T., M. The Delphi Method. Techniques and Applications. Addison-Wesley, 1975.

26. MAÑAS, J. A. Prueba de Programas España: [Consultado el: 28 de marzo de 2007]. Disponible en: http://www.lab.dit.upm.es/%7Elprg/material/apuntes/pruebas/testing.htm.

27. ---. Pruebas de Programas [Consultado el: 8 de febrero del 2007 Disponible en: http://www.upv.es/protel/usr/jotrofer/pascal/testing.htm.

28. MORENO, G. A. Pruebas Granada: [Consultado el: 28 de marzo de 2007]. Disponible en: http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/.

29. MYERS, B. A. A new model for handling input. ACM Transactions on Information Systems 1979.

30. ---. A new model for input. ACM Transactions on Information Systems, 1990.

31. PARETO, D. D. Diagrama de Paretto. El Diagrama de Pareto es una gráfica en donde se organizan diversas clasificaciones de datos por orden descendente, de izquierda a derecha por medio de barras sencillas después de haber reunido los datos para calificar las causas. De modo que se pueda asignar un orden de prioridades. p.

32. PATÓN, D. E. F.-M. INGENIERÍA DEL SOFTWARE I España: Universidad de Castilla-La Mancha, [Consultado el: 28 de abril de 2007]. Disponible en: http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf.

33. PÉREZ, H. ¿Qué es el testeo? España: [Consultado el: 12 de febrero de 2007]. Disponible en: http://www.astest.com.ar/spip/spip.php?article10.

34. ---. "Testeo de Software de Aplicación "España: Última actualización: 26 de enero del 2007 [Consultado el: 15 de febrero de 2007]. Disponible en: http://www.astest.com.ar/spip/spip.php?article9

35. ---. "Testeo de Software" España: Última actualización: 26 de enero del 2007 [Consultado el: 16 de febrero de 2007]. Disponible en: http://ar.groups.yahoo.com/group/testeosoftware/

36. PÉREZ, Y. F.; DELGADO, R. L. C., et al. Proceso de prueba de aceptación para un software de gestión Cuba: Sin publicar, de 6 de febrero del 2007].

37. PRADO, O. "Impacto sobre métodos, práctica y herramientas de testeo" [Pagina Web]. España: Última actualización: 23 de mayo de 2006. [Consultado el: 9 de marzo de 2007]. Disponible en: http://www.astest.com.ar/spip/spip.php?article6

38. ---. "Técnicas de testeo" [Pagina Web]. España: Última actualización: 26 de enero del 2007 [Consultado el: 10 de marzo de 2007]. Disponible en: http://www.astest.com.ar/spip/spip.php?rubrique1.

39. ---. "A través del ciclo de vida" [Pagina Web]. España: Última actualización: 26 de enero del 2007 [Consultado el: 9 de marzo de 2007]. Disponible en: http://www.astest.com.ar/spip/spip.php?rubrique11.

40. PRESSMAN, R. Ingeniería de Software, Un enfoque practico. Quinta ed. España: Mc Graw-Hill, 2002.

41. ---. Ingeniería de Software. Un enfoque Práctico. España: McGraw- Hill, 1998.

42. PRESSMAN, R. S. Ingeniería de Software. Un enfoque práctico. 4 edición ed. España: Mc Graw- Hill, 2000.

43. QUESADA, J. A. L. Pruebas de Software [Artículo Digital]. España: [Consultado el: 5 de diciembre de 2006]. Disponible en: http://dis.um.es/~lopezquesada.

44. RICE, R. W. "Surviving the top 10 challenges of software test automation". Cross Talk: The Journal of Defense Software Engineering, 2002. 26-29 p.

45. RUP, R. S. C. Ayuda Ampliada de RUP. Estados Unidos: 2003,

46. SCOTT, W. Métodos de Prueba Orientada a Objetos para el ciclo de Vida Completo (FLOOT) Última actualización: 27 de enero del 2007. [Consultado el: 8 de febrero del 2007 Disponible en: http://www.ambysoft.com/essays/flootSpanish.html.

47. SOTO, M. [Pagina Web]. España: Última actualización: 18 de mayo de 2006. [Consultado el: 10 de marzo de 2007]. Disponible en: http://www.astest.com.ar/spip/spip.php?article5.

48. SPILLNER, A. The W-MODEL – Strengthening the Bond Between Development and Test Germany: University of Applied Sciences Bremen, [Consultado el: 28 de marzo de 2007].

49. TERUEL, A. El Plan de Prueba Última actualización: 20 de enero del 2001. [Consultado el: 8 de febrero del 2007 Disponible en: http://www.ldc.usb.ue/~teruel/ci4713/clases2001/planPrueba.html.

50. ---. Requerimiento de Prueba Última actualización: 11 de enero del 2001. [Consultado el: 8 de febrero del 2007 Disponible en: http://www.ldc.usb.ue/~teruel/ci4713/clases2001/testSpecs.html.

51. TERUEL., P. A. Las Pruebas de Verificación de Requerimientos [Página Web]. Venezuela: Universidad Simón Bolívar de Venezuela, Última actualización: 4

de enero de 2001. [Consultado el: 15 de diciembre de 2006]. (Ingeniería de Software III). Disponible en: http://www.ldc.usb.ue/~teruel/.

52. USAOLA, D. M. P. Mantenimiento Avanzado de Sistemas de Información Pruebas del Software [Consultado el: 18 enero 2007 Disponible en: http://alarcos.inf-cr.uclm.es/doc/masi/doc/lec/parte5/polo-apuntesp5.pdf.

53. WIKIMEDIA FOUNDATION, I. “Pruebas de software” [Pagina Web]. Última actualización: 18 enero 2007 [Consultado el: 25 de febrero de 2007]. Disponible en: http://es.wikipedia.org/wiki/Pruebas_de_software.

Anexos

Anexo 1 Formato de Prueba de Caja Blanca

Nombre de la empresa

Nombre del Sistema: _____ Fecha: _____

Descripción de la prueba_____

Módulo_____

Proceso: _____

Descripción del proceso: _____

Objetivo de la prueba:_____

Nombre del dato	Dato entrada	Proceso	Salida esperada	Resultado		Observaciones

Persona que realizó la prueba: _____

Persona que revisó la documentación: _____

NOTA: En caso necesario anexar hoja de detalle de las pruebas y observaciones, por cada modulo del sistema se debe llenar un formato.

Anexo 2 Formato de Prueba de Caja Negra

Nombre de la Empresa: _____

Nombre del sistema: _____

Tipo de prueba: _____

Fecha: _____

Módulos	Submodulos	Parámetros				Resultados		Observaciones
		1	2	3	4	Pasa	No pasa	

Listas de parámetros:

Nota: Ninguna de las dos estrategias de pruebas es ideal, sin embargo la prueba de especificación es la más eficiente, ya que se centra en la forma que se espera que se use el software.

Anexo 3 Plantilla de un Plan de Pruebas

< Nombre del proyecto >

Versión < número versión>

Revisión histórica

Fecha	Versión	Descripción	Autor

Índice

1. Introducción

1.1. Propósito

1.2. Ámbito

1.3. Definiciones, acrónimos y abreviaturas.

2. Requerimientos de las pruebas

3. Estrategia de prueba

3.1. Tipos de pruebas y técnicas

Objetivos de la prueba	
Técnicas	
Criterios de finalización	
Consideraciones	

3.2. Herramientas

Tipo de Prueba	Herramienta	Versión

4. Recursos

4.1. Recursos hardware

Recurso	Cantidad	Nombre y Tipo

4.2. Recursos software

Nombre del elemento software	Versión	Tipo y otras notas

4.3. Herramientas de soporte

Categoría de herramienta o tipo	Marca de la herramienta	Comprada o de la empresa	Versión

4.4. Configuración de entornos de prueba

Nombre de configuración	Descripción	Implementación y configuración física

4.5. Recursos humanos

RECURSOS HUMANOS		
Rol	Mínimos recursos recomendados	Responsabilidades específicas o comentarios

5. Actividades de prueba

Actividad	Esfuerzo (p/d)	Fecha de comienzo	Fecha de finalización

6. Resultados de las pruebas

Documento de desarrollo de software	Desarrollador	Revisión	Fecha

7. Tareas de la etapa de prueba

Anexo 4 Encuesta sobre el Flujo de Trabajo de Prueba a asesores de calidad de los proyectos

Nombre del encuestado: _____

Ocupación: _____

1- ¿Cuántos proyectos de gestión se desarrollan en su facultad? _____
2- ¿En cuántos de ellos se está llevando a cabo un proceso de pruebas? _____
3- ¿Cuántos de estos proyectos planifican sus pruebas? _____
4- ¿Cuántos proyectos cuentan con un repositorio de pruebas? _____
5- ¿En cuántos de ellos se analizan los resultados de las pruebas? _____
6- ¿Cuántos proyectos tienen trazada y siguen una estrategia de prueba? _____
7- ¿Cuántos de ellos siguen algún estándar para llevar a cabo el proceso de pruebas? _____
    a. Mencione él o los estándares_____
8- ¿Cuántos proyectos en su facultad aplican métodos de pruebas de :

___Caja Blanca (Pruebas que se basan en los detalles procedimentales)

__ Camino Básico
__ Condición
__ Bucles
__ Flujo de datos

___Caja Negra (Se llevan a cabo sobre la interfaz de software)

__ Partición Equivalente
__ Análisis del Valor Limite

9- ¿Cuántos proyectos realizan los siguientes tipos de pruebas:

_____Funcionalidad
_____Seguridad
_____Volumen
_____Usabilidad
_____Integridad
_____Stress
_____Rendimiento (Carga)
_____Configuración

_____Instalación

Otras_____

10- ¿En cuántos proyectos de su facultad se desarrollan estos niveles de pruebas?

_____ Unidad (proceso de verificación en la menor unidad del diseño)

_____Integración (para construir la estructura del programa)

_____Sistema (integración del software en el sistema)

_____Regresión (no se introduzcan errores adicionales con los cambios)

_____Aceptación (Verificar que el software este listo)

11- ¿En cuántos de sus proyectos se desarrollan las pruebas?:

_____ Durante todo el ciclo de vida del proyecto

_____ Al tener una versión del sistema

_____Al producto final

12- ¿En cuántos de sus proyectos se utiliza alguna herramienta que facilite el trabajo del flujo de pruebas? _____

a. ¿Cuáles? _____

13- ¿En cuántos proyectos exigen documentación del flujo de trabajo de pruebas?

___ Plan de pruebas

___ Descripción de los Casos de Uso

___ Documento de arquitectura

___ Estrategia de pruebas

___ Resultados de las pruebas anteriores

___ Documento de no Conformidades

___ Manual de usuarios

¿Otros? _____

a. ¿Cuáles? _____

14- ¿En cuántos proyectos de su facultad se le da capacitación al equipo de pruebas para desarrollar exitosamente este proceso? _____

Anexo 5 Encuesta a Directivos de Calidad de la Universidad

Nombre del encuestado: _____
Ocupación: _____

1. Metodologías que más comúnmente se utilizan para el desarrollo de software en la universidad.
   ___RUP
   ___XP
   ___FSM
   Otras_____

2. ¿Considera que por lo general los software que se liberan en la universidad las expectativas de los clientes?
   Sí__ No__

3. Atributos de calidad que cree que cumplen los productos liberados en la Universidad:
   ___Confiabilidad
   ___Seguridad
   ___Interfaz
   Otros_____

4. ¿Se lleva a cabo en el de desarrollo de software en la universidad algún proceso de pruebas en el ámbito del proyecto?
   Sí __ No__

5. ¿En que Niveles de Pruebas se están desarrollando las pruebas a software?
   ___ Unidad (proceso de verificación en la menor unidad del diseño)
   ___ Integración (para construir la estructura del programa)
   ___ Sistema (integración del software en el sistema)
   ___Regresión (no se introduzcan errores adicionales con los cambios)
   ___ Aceptación (Verificar que el software este listo)

6. ¿Qué Métodos de Prueba son aplicados?:
   ___Caja Blanca (Pruebas que se basan en los detalles procedimentales)
   ___Caja Negra (Se llevan a cabo sobre la interfaz de software)

___Pruebas de Estructuras de control

Otros_____

7. ¿Qué tipos de pruebas a software se realizan más comúnmente?

__ Camino Básico

__ Condición

__ Partición Equivalente

__ Análisis del Valor Limite

Otras_____

8. ¿Por lo general se planifican las pruebas a software en el ámbito del proyecto?

Si____ No____

9. ¿Cuentan los proyectos con un repositorio de pruebas?

Si____ No____

10. ¿Se analizan los resultados de las pruebas aplicadas en los proyectos de la universidad?

Si____ No____

11. ¿Como calificaría el estado de las pruebas de software en la universidad?

Adecuada _____ Media_____ Insuficiente_____

12. En caso de no ser adecuada explique las causas que considera provocan esto.

__Falta de una adecuada estructura organizativa en las empresas para este efecto.

__Desconocimiento del tema o técnicas a aplicar.

__Escaso personal capacitado y disponible para ello.

Otras_____

13. ¿Utilizan los proyectos de la universidad algún software para realizar las pruebas a software?

Si____ No____

14. ¿Utilizan los proyectos de la universidad algún software para organizar el proceso de pruebas?

Si____ No____

Anexo 6 Datos de los Expertos que conforman el Panel

Experto	**Graduado de**	**Años vinculados a la UCI / Softel**	**Ocupación**	**Eventos científicos**	**Experiencia en el tema de pruebas.**
Experto 1	Ingeniería en informática	4 años	Especialista Dirección de Calidad y Normas de la UCI.	➢ Curso avalado de CMMI ➢ Evento Sepgla 2004 ➢ Evento Metánica de calidad ➢ Informática 2007 ➢ Evento Uciencia	➢ Registros y Notarías ➢ Identidad ➢ Digitalización ➢ Multimedia en General ➢ Intranet de PDVSA ➢ Pruebas de aceptación en el extranjero.
Experto 2	Ingeniería en Informática	2 años	Profesor y asesor de Calidad de la Facultad 8	➢ Forum de Ciencia y técnica	➢ A 40 años de una idea ➢ Constitución II ➢ Proyectos de Ingles ➢ Historia Universal ➢ Contenidos ➢ Portales ➢ Libro Electrónico ➢ Multimedia de Fidel Castro ➢ Identidad ➢ Registro y Notaría
Experto 3	Licenciada en Cibernética y Matemática	19 años en la empresa Softel	Especialista en la Dirección de Desarrollo de Softel.	➢ Talleres de Informática de Calidad 2003, 2005, 2007. ➢ Uciencia. ➢ Eventos Científicos CIMEQ. ➢ Eventos de las Brigadas Técnicas Juveniles.	➢ Sistema de Cuidados Intensivos CIMEQ. ➢ Sistemas Estadísticos. ➢ Sistema Modular Gestión Hotelera. ➢ Sistema Informatización SIME ➢ Sistema Informatización

					MINTUR. ➢ Sistema de Dirección de Cuadros de Grupo de Electrónica. ➢ Dirección de Calidad de Softel.
Experto 4	Ingeniería en Informática	4 años	Especialista de la Dirección de Calidad y Normas	➢ Evento Uciencia. ➢ Evento Informática 2007	
Experto 5	Graduada de Ingeniera en Sistema Automatizado de Dirección.	5 años	Profesora de Ingeniería de Software de la UCI.	➢ Evento Uciencia ➢ Evento Informática 2007 ➢ Curso avalado de CMMI	
Experto 6	Ingeniería Informática	2 años	Profesor de Ingeniería de Software de la UCI.	➢ Informática 2005 ➢ Informática 2007	➢ Proyecto CICPC (Casos de Estudio).

Anexo 7 Encuesta a Miembros del Panel de Expertos

Nombre del Encuestado: _____

Ocupación: _____

1. ¿Considera usted que es importante llevar a cabo un adecuado proceso de pruebas en los proyectos de producción de software de gestión en la Universidad, para asegurar la calidad de los procesos de desarrollo y de los productos que se obtengan de los mismos?:

   ____Sí ____No ____No Sé

   ¿Por qué?

   _____

   _____

   ____ _____

2. De una valoración de 1a 5 (siendo 5 el mayor valor) de los criterios expuestos a continuación según la propuesta a validar:

   ____Satisfacción de las necesidades de los probadores

   _____Necesidad del empleo de la propuesta

   _____Adaptabilidad a proyectos productivos, independientemente de la metodología que se utilice.

   _____Adaptabilidad a cualquier herramienta que facilite el flujo de trabajo de pruebas.

   _____Repercusión en la calidad del producto.

   _____Impacto en las pruebas de aceptación del cliente.

   _____Posibilidad de aplicación.

3. ¿Cuáles considera usted que sean las principales acciones que se deban llevar a cabo en los proyectos de producción de software de gestión de la UCI para llevar a cabo un adecuado proceso de pruebas?:

   1)

   2)

   3)

   4)

5)
6)
7)
8)
9)
10)

4. ¿Considera usted que se pueda aplicar exitosamente en la universidad el proceso propuesto, de manera que permita mejorar la calidad de los procesos y productos que se desarrollan en nuestros proyectos productivos, y que se le daría cumplimiento a su objetivo?:

   ____Sí ____No ____No Sé

   ¿Por qué?

   _____

   _____

   ____ _____

5. ¿Qué factores cree usted que podrían atentar contra la correcta aplicación de este proceso en la UCI, y cuales lo facilitarían?

   _____

   _____

   ____ _____

6. Haga un comentario o aporte sobre el proceso usted está evaluando. (El comentario es libre y debe reflejar algún elemento de interés que aporte elementos a la mejora del proceso):

   _____

   _____

   ____ _____

Anexo 8 Plantilla de No Conformidades

<Nombre del Proyecto>
Módulo <Nombre del Módulo>

Registro de versiones:

Fecha	Versión	Descripción	Autor
<00/00/0000>	<1.0>	<Descripción>	<Autor>

Aspectos generales

Descripción de Aspectos Generales a tener en cuenta a la hora de analizar el resultado de las pruebas, incidencias en el momento de su desarrollo y otros aspectos relevantes.

Elementos Probados

Descripción general o lista de los Elementos Probados, y otros aspectos importantes a tener en cuenta a la hora analizar las No Conformidades Detectadas.

Elementos No Probados y causas

Descripción general o lista de los Elementos no Probados, la causa de que no se hayan podido realizar las Pruebas y cualquier otro elemento importante que aporte la información necesaria para que sean analizadas estas causas y resueltas para la siguiente iteración.

Tabla de No Conformidades Detectadas

Elemento	No	No Conformidad	Aspecto Correspondiente	Etapa de detección	Importancia	Recomendación
Nombre de Elemento	<X>	<Descripción de la No Conformidad>	<Descripción de Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>

Recomendaciones

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección
<Nombre de Elemento>	<X>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de la detección del error>

Anexos

Anexo <1>

Anexo <2>

Anexo <3>

Glosario de Términos

Arquitectura de automatización de pruebas: proporciona una descripción arquitectónica del sistema de automatización de pruebas, usando un número de diversas vistas arquitectónicas para representar diversos aspectos del sistema.

Artefactos: resultados tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Ciclo de Vida del Software: conjunto de fases que agrupan disímiles actividades con el objetivo de obtener un producto final. Facilita el control sobre los tiempos en que es necesario aplicar recursos de todo tipo (personal, equipos, suministros, etc.) al proyecto.

Configuración de ambiente de pruebas: es la especificación para un arreglo del hardware, software, y características asociadas al ambiente que se requieren para hacer posible la efectividad de las pruebas que conducirán a evaluar uno o varios objetivos o blancos a probar.

Datos de pruebas: es la realización o implementación de la prueba, donde los scripts de prueba incorporan los aspectos procedimentales, y los datos de prueba definen características. Es una colección de valores de entrada de pruebas que se consumen durante la ejecución de una prueba, y los resultados previstos referenciados para los propósitos comparativos durante la ejecución de una prueba.

Especificación de interfaz de prueba: define un sistema requerido de comportamientos hechos por un clasificador (específicamente, una clase, un subsistema o componente) para los propósitos de la prueba. Los tipos comunes incluyen el acceso a pruebas, el comportamiento tropezado, diagnostico de registro y oráculos de las pruebas. Cada interfaz de la prueba debe proporcionar un grupo único y bien definido de servicios.

Flujo de Trabajo: secuencia de actividades que van a realizar un grupo de personas para obtener diferentes artefactos.

Iteración: es una secuencia de actividades con un plan establecido y criterios de evaluación, cuyo resultado es una versión del software.

Lista de Ideas de Pruebas: Una lista enumerada de las ideas, formada a menudo parcialmente, que identifican pruebas potencialmente útiles para conducir.

Logs de pruebas: es una colección de salida cruda capturada durante una ejecución única de una o más pruebas, representando generalmente la salida, resultado de la ejecución de una prueba para un solo funcionamiento del ciclo de la prueba.

Métricas: se refieren a un amplio rango de medidas para el software dentro del contexto de la planificación del proyecto de software.

Modelo: es una abstracción del sistema, especificando el sistema desde un punto de vista y un determinado nivel de abstracción.

Plan de prueba: es la definición de las metas y de los objetivos de la prueba dentro del alcance de la iteración (o del proyecto), los elementos que son registrados, las medidas que se tomarán, los recursos requeridos y los entregables que se producirán.

Plan de Iteración: define las principales metas, objetivos, motivaciones, medidas, recursos, un horario y entregables de prueba durante cada iteración.

Proceso de Desarrollo de Software: es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

Repositorio de pruebas: lugar donde se almacena toda la documentación referente a las pruebas, pueden ser planillas, documentos de no conformidades. Este repositorio se recomienda que sea central para la utilización general de todos.

Resumen de evaluación de pruebas: organiza y presenta un análisis resumido de los resultados de la prueba y de las medidas de la prueba para la revisión y el asesoramiento, típicamente por los participantes. Además, puede contener una declaración general de la calidad relativa y proporcionar las recomendaciones para el esfuerzo futuro de la prueba.

Script de pruebas: representa las instrucciones paso a paso para realizar una prueba, permitiendo su ejecución. Puede tomar la forma de cualquier instrucción documentada textual que se ejecutan manualmente o las instrucciones legibles por computadora que permiten la ejecución automatizada de la prueba.

Software de Gestión: son software que realizan procesamientos de datos e información comercial y además realizan cálculos iterativos.

SPSS *(Statistical Product and Service Solutions*): es un programa estadístico informático muy usado en las ciencias sociales y empresas de investigación de mercado para el análisis estadístico de datos. Originalmente SPSS era el acrónimo de (*Statistical Package for the Social Sciences*).

Suite de pruebas: conjunto de herramientas integradas para ejecutar las actividades o un grupo de actividades del flujo de trabajo de prueba de forma automática.

Sumario de evaluación de prueba: organiza y presenta un análisis sumario de los resultados de la prueba y de las medidas de la prueba para la revisión y el asesoramiento, típicamente por los participantes o involucrados. Además, el resumen de la evaluación de la prueba puede contener una declaración general de la calidad relativa y proporcionar las recomendaciones para el esfuerzo futuro de la prueba.