PEDECIBA informática
Instituto de Computación (InCo)
Facultad de Ingeniería
Universidad de la República.

Tesis de Maestría en Informática

PROCESO DE TESTING FUNCIONAL INDEPENDIENTE

BEATRIZ PÉREZ LAMANCHA

Supervisora y Orientadora de Tesis: Dra. Nora Szasz

Montevideo, Uruguay

2006

PÁGINA DE APROBACIÓN

FACULTAD DE INGENIERÍA

El tribunal docente integrado por los abajo firmantes aprueba la Tesis de Maestría en Informática:

Título: Proceso de Testing Funcional Independiente

Autor: Beatriz Pérez Lamancha

Supervisora y Orientadora de Tesis: Dra. Nora Szasz

Maestría en Informática

Fallo del Tribunal: Excelente

Tribunal

Dra. Juliana Herbert
Dr. Raul Ruggia
MSc. Omar Viera

Fecha: 21 de Setiembre de 2006

RESUMEN

El proceso de desarrollo de software describe la vida de un producto de software desde su concepción hasta su entrega, utilización y mantenimiento. En forma análoga, el proceso de prueba describe la forma en que el producto de software debe ser probado.

El proceso de prueba puede ser visto como parte del proceso de desarrollo de software o independiente de él. En este último caso, el proceso de prueba no tiene en cuenta la forma en que se realiza el desarrollo para definir las actividades a realizar.

En este trabajo se define un proceso para la prueba funcional de un producto de software, independiente del proceso seguido para su desarrollo. La prueba funcional de un producto de software tiene como objetivo validar cuando el comportamiento observado del producto cumple o no con sus especificaciones. El proceso definido se llama ProTest.

El Centro de Ensayos de Software (CES) es una organización que se dedica a las pruebas independiente de productos de software. La motivación para este trabajo surge de la necesidad de definir la metodología de trabajo a utilizar para realizar pruebas funcionales independientes en el Laboratorio de Testing Funcional del CES.

Con el fin de definir ProTest, se realiza un estudio del estado del arte en lo referente a las pruebas y al proceso de pruebas de software. Esta información es organizada y resumida para la realización de las pruebas funcionales de un producto.

Se presentan las etapas definidas para ProTest, las actividades, artefactos y roles. Se describe la aplicación del proceso definido en un caso de estudio: el proyecto de prueba de una aplicación de gestión. Se presentan las conclusiones y ajustes a partir de dicha experiencia.

Se concluye que el proceso es una guía útil para realizar pruebas funcionales de productos de software. Actualmente, Protest es usado en los proyectos de pruebas del Laboratorio de Testing Funcional del Centro de Ensayos de Software. Para cada proyecto de prueba, el proceso es adaptado a las características del proyecto. Al culminar el mismo, se evalúa ProTest y se realizan las mejoras, a partir de lo aprendido con la experiencia.

Palabras Clave: Proceso de Pruebas, Pruebas de Software, Pruebas Funcionales, Pruebas Independientes.

AGRADECIMIENTOS

A Nora por alentarme a continuar, por su comprensión, cariño, paciencia y dedicación.

Al PEDECIBA, al In.Co. y al CES. A la Comisión Académica de Posgrado de la Facultad de Ingeniería por becarme para poder realizar este trabajo.

A mis compañeros del Grupo de Ingeniería de Software, por alivianar mis tareas de enseñanza mientras terminaba la tesis.

A mis compañeros del Centro de Ensayos de Software. Es un honor para mí trabajar con ustedes. Son excelentes profesionales y mejores personas. Gracias a todos por su apoyo y colaboración en este trabajo.

Al cliente que me permitió compartir como caso de estudio de esta tesis la experiencia de trabajo con él.
A Mariana por su dedicación y colaboración.

A mis compañeros de Proyecto de Ingeniería de Software, porque disfrutamos lo que hacemos. A los estudiantes del Grupo 1 de Proyecto de Ingeniería de Software del año 2005 y del Grupo 3 del año 2004 por haber desarrollado las herramientas DeVeloPro y Graphead.

A mis compañeros de oficina y alrededores en el InCo, por hacer el ambiente de trabajo ameno y divertido. Gracias por la paciencia y por sacarme una sonrisa en los malos momentos, que no fueron pocos.

A mis amigos, compañeros de la vida: Andrea, Ana, Doris, Martín, Rodrigo.

A mi familia: Mamá, Papá, Abuela, Carlos, Marga, Irene, Rodrigo, Ana, Antonia, Luis y Tomás. Los llevo en el corazón, siempre.

"El que tenga una canción tendrá tormenta
El que tenga compañía, soledad
El que siga un buen camino tendrá sillas
Peligrosas que lo inviten a parar
Pero vale la canción buena tormenta
Y la compañía vale soledad
Siempre vale la agonía de la prisa
Aunque se llene de sillas la verdad"

Silvio Rodríguez

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1

INTRODUCCION

Desde el punto de vista del cliente y los usuarios, la calidad de un producto de software es percibida principalmente por las fallas que encuentran en el producto y por la gravedad que éstas tienen para el negocio del cliente. Para ser competitivas, las empresas desarrolladoras de software necesitan asegurarse de la calidad de sus productos previo a su instalación en el ambiente del cliente.

La prueba exhaustiva del producto antes de ser entregado al cliente implica probar el comportamiento del mismo para todas las combinaciones válidas e inválidas de entradas, bajo cada estado posible del sistema. Esto, incluso para un programa pequeño puede llevar cientos de años y es económicamente inviable. Debido a esto, las empresas que desarrollan software intentan una solución de compromiso, esto es entregar sus productos con la menor cantidad de defectos posible. Para esto, definen cuando entregar el producto a los usuarios en función del costo y el beneficio de realizar las pruebas.

El desafío está en encontrar los defectos que tendrían mayor impacto negativo para el negocio del cliente antes de que el producto se haya entregado. Para lograr este objetivo, existen técnicas y estrategias para definir y ejecutar las pruebas y es necesario contar con un equipo entrenado en ellas. Este equipo puede ser parte de la organización de desarrollo o puede ser externo. Ambas formas tienen sus ventajas y desventajas.

El Centro de Ensayos de Software (CES) es una organización que se dedica a las pruebas independientes de productos de software. La motivación para este trabajo surge de la necesidad de definir la metodología de trabajo a utilizar para realizar pruebas funcionales independientes en el CES.

Para realizar pruebas funcionales en forma independiente, el proceso a definir no puede depender de la metodología usada para el desarrollo y debe estar enfocado en realizar pruebas funcionales sobre una versión ejecutable del producto, sin contar con el código fuente. De este modo, la organización que desarrolla el producto no necesita brindar su principal capital, que es el código fuente, para contratar los servicios que brinda el CES.

El proceso definido será utilizado en los proyectos de pruebas funcionales que realice el CES. Por ello, la documentación del proceso debe estar accesible a todos los integrantes del equipo de prueba. En cada proyecto donde el proceso es utilizado, pueden surgir ajustes a realizar, para mejorarlo o extenderlo. Un cambio posible puede ser la mejora de la descripción de una actividad, pero también un cambio puede ser que para determinados tipos de proyectos de prueba, se realice una actividad extra, extendiendo así el proceso de prueba. La documentación además de ser accesible, debe ser fácil de ajustar, para crear nuevas versiones del proceso.

En la sección 1.1 de este capítulo se presenta la motivación de este trabajo: las pruebas independientes y el Centro de Ensayos de Software, en la sección 1.2 se describen las principales contribuciones de esta tesis y en la sección 1.3 se explica la organización del resto del documento.

1.1 Motivación: Independencia de las Pruebas y el CES

La falta de objetividad que ocurre cuando el equipo de desarrollo es el mismo que el que realiza las pruebas sumado a las presiones institucionales por salir al mercado hace que en muchos casos la calidad del software se conozca recién cuando se pone en producción.

Las ventajas que se obtienen al tener una organización de prueba independiente de las de desarrollo son: motivación en el proceso de pruebas, oposición de intereses con la organización de desarrollo, separación del proceso de pruebas del control gerencial de la organización de desarrollo y el conocimiento especializado que la organización independiente tiene respecto a las pruebas [Mey04].

En organizaciones de pequeño y mediano porte la subcontratación de los servicios de prueba es imprescindible para implementar la prueba independiente, ya que es difícil sustentar la existencia de un área dedicada exclusivamente a las pruebas dentro de la organización misma.

En Uruguay la industria de software está compuesta en más de un 90% por pequeñas empresas y ha presentado en los últimos años un gran dinamismo. Debido al tamaño reducido del mercado interno, esta se ha volcado al mercado internacional de forma que Uruguay llegó a constituirse en el año 2001 en el principal exportador de software de América Latina [Sto03].

En el año 2004 se crea el Centro de Ensayos de Software (CES) [CES06]. Se trata de un emprendimiento conjunto de la Universidad de la República de Uruguay (UdelaR), a través de la Fundación Julio Ricaldoni, y de la Cámara Uruguaya de Tecnologías de la Información (CUTI), entidad que agrupa a la mayoría de las empresas productoras de software del país.

Los servicios que ofrece el CES incluyen:

- Servicios de prueba independiente: Planificar, diseñar, coordinar y ejecutar pruebas de productos de software de manera efectiva y controlada, definiendo claramente el contexto y los objetivos.
- Consultoría: Asesorar a las organizaciones en la mejora de los procesos de prueba, definición de estrategias y automatización de las pruebas. Colaborar en la creación y consolidación de sus áreas de prueba.
- Capacitación: Elaborar e impartir programas de capacitación en la disciplina de testing según las necesidades de cada organización.

El CES se compone de dos laboratorios: el Laboratorio de Testing Funcional (LTF) enfocado en la evaluación de productos desde el punto de vista funcional y el Laboratorio de Ensayos de Plataformas (LEP), donde se realizan pruebas de desempeño y se asiste a la industria para resolver problemas de funcionamiento en arquitecturas de hardware y software complejas.

La creación del CES es una parte del proyecto "Desarrollo tecnológico en sectores clave de la economía uruguaya", que cuenta con financiación de la Unión Europea durante los tres primeros años, luego de los cuales el CES debe ser auto sustentable. También cuenta con aportes de la CUTI y la UdelaR y contó con financiación del Programa de Naciones Unidas para el Desarrollo (PNUD) para realizar actividades preparatorias previas a la aprobación del proyecto por la Unión Europea [Tri04].

Dentro de las actividades definidas en el proyecto de creación del CES para el Laboratorio de Testing Funcional se encuentra la de "Definición de procesos y procedimientos". Se consideró oportuno enmarcar dicha actividad en el programa de formación académica. Esta maestría es el resultado obtenido del estudio realizado para la definición de los procesos y procedimientos a utilizar.

1.2 Contribución

Las principales contribuciones de este trabajo pueden agruparse en tres áreas las cuales están estrechamente relacionadas.

El primer aporte es el estudio del estado del arte en lo referente a las pruebas de software y al proceso de pruebas de software. Se estudian las definiciones, los elementos, las técnicas y estrategias existentes para la realización de las pruebas de software. Con el fin de conocer las buenas prácticas ya probadas referentes a las pruebas de productos de software, se estudian los procesos y metodologías de pruebas existentes. Esta información es organizada y resumida en este trabajo con el objetivo de poder ser utilizada como referencia para la realización de las pruebas funcionales de un producto.

Los distintos autores definen un ciclo de vida específico para las pruebas, que, en general, separa en distintas etapas las actividades de planificación, diseño y las de ejecución de las pruebas. Edward Kit [Kit95] define las etapas del proceso de prueba y sus principales actividades. Rex Black [Bla02] se focaliza en la planificación de las pruebas, definiendo las fases sin entrar en detalle en sus actividades. Cem Kaner, James Bach y Bret Pettichord [KBP01] recopilan 293 lecciones aprendidas sobre las pruebas, sin organizarlas en un proceso. Hetzel [Het88] define las fases de prueba acompañando el ciclo de vida del software. Cem Kaner, Jack Falk y Hung Quoc Nguyen [KFN99] describen las técnicas de prueba que son útiles en cada etapa del ciclo de vida del software, sin agruparlas en fases o definiendo actividades específicas. James A. Whittaker [Whi00] y International Software Testing Qualifications Board (ISTQB) [ISQ05] definen las fases del proceso de prueba, sin entrar en el detalle de las actividades que las componen. TMap [PTV02] es el proceso de prueba definido por la empresa Sogeti, define las fases y el detalle de las actividades de prueba.

El segundo aporte constituye la contribución principal del trabajo: la definición de una metodología que sirve como guía para realizar pruebas funcionales independientes de un producto de software en el Laboratorio de Testing Funcional del CES.

El definir dicha metodología incluye definir las actividades específicas para realizar las pruebas de un producto de software y las actividades para definir la propuesta de servicio, la planificación y la gestión del proyecto de prueba en forma independiente del proyecto de desarrollo.

Se define un proceso de prueba funcional llamado ProTest basado en aquellas buenas prácticas propuestas en la literatura estudiada. Estas propuestas fueron adaptadas para ser usadas en una evaluación independiente. A diferencia de las otras propuestas, los ciclos de prueba en ProTest son lo que guían el proceso de prueba.

ProTest fue puesto en práctica en un proyecto concreto del CES. Se describe este caso de estudio y se presentan las conclusiones de su aplicación. Asimismo de esta primera experiencia surgieron ajustes que debieron realizarse al proceso luego de su puesta en práctica.

ProTest es actualmente el proceso usado en el Laboratorio de Testing Funcional del CES, el cual es ajustado y mejorado con cada proyecto de prueba realizado en el Laboratorio. Ha sido presentado en distintos eventos, los que se describen a continuación:

- ”Metodología para Verificación Independiente" Resumen presentado en las IX Jornadas de Informática e Investigación Operativa, Facultad de Ingeniería de la Universidad de la República, Montevideo, Uruguay. Noviembre de 2004.
- “ProTest – Proceso de Testing Funcional" Presentación en las V Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC’ 06), Puebla, México, Febrero de 2006 [Per06].

- "Metodología usada en el Laboratorio de Testing Funcional del Centro de Ensayos de Software" Resumen presentado en las I Jornadas de Testing del Centro de Ensayos de Software, Hotel NH Columbia, Montevideo, Uruguay. Mayo de 2006 [CES06].

El tercer aporte surge de la necesidad de contar con una herramienta para documentar y gestionar el proceso definido. Al momento de comenzar esta tesis, no existían herramientas gratuitas que permitan realizar este tipo de actividades y es por este motivo que se plantea la construcción de una.

Se definieron los requerimientos que debe satisfacer esta herramienta y se realizó el seguimiento de su construcción cumpliendo el rol de cliente, validando los distintos prototipos de la herramienta generada. Entre los requerimientos definidos para la herramienta se encuentran: posibilidad de definir los distintos elementos de un proceso (actividades, roles, artefactos, fases, definiciones y herramientas), posibilidad de administrar distintas versiones de un proceso permitiendo conocer el elemento que cambió, cual es el motivo del cambio y quién lo realizó, posibilidad de generar la documentación en forma automática para cada versión de un proceso en dos formatos: un sitio web con páginas html estáticas y un documento en pdf.

El desarrollo de la herramienta fue realizado por un grupo de estudiantes. Permite definir tanto procesos de desarrollo de software como procesos de prueba. La herramienta se llama DeVeloPro y es utilizada como soporte documental de ProTest, generando en forma automática un sitio web y un documento de texto imprimible [DVP06].

1.3 Organización del Documento

En el Capítulo 2 se brinda una visión general de las Pruebas de Software, definiendo los principales términos que serán usados en esta tesis, los distintos tipos de pruebas y las técnicas existentes.

En el Capítulo 3 se describe la relación entre el proceso de pruebas y los procesos de desarrollo, los modelos de calidad y modelos de mejora.

En el Capítulo 4 se presenta ProTest, proceso propuesto para la prueba funcional de un producto de software. Se presentan las principales características del proceso de prueba, brindando una visión global de sus etapas, actividades, artefactos y roles.

En el Capítulo 5 se describen las actividades definidas para ProTest, organizadas según sus etapas.

En el Capítulo 6 se describe el contenido de los artefactos definidos en ProTest y los roles del proceso, para cada uno se muestran los requisitos y competencias, las actividades en las que participa y las responsabilidades en cada actividad.

En el Capítulo 7 se presenta un caso de estudio: la aplicación de ProTest en un proyecto de prueba del Laboratorio de Testing Funcional del CES. Se describe cómo fueron seguidas las etapas definidas para la prueba de un sistema de gestión.

En el Capítulo 8 se detallan las conclusiones de esta tesis y los posibles trabajos a futuro.

En el Anexo A, se presenta DeVeloPro, herramienta para la documentación y administración de procesos.

Capítulo 2

PRUEBAS DE SOFTWARE

En este capítulo se brinda una visión general de las Pruebas de Software, definiendo los principales términos que serán usados en esta tesis, los distintos tipos de pruebas y las técnicas existentes.

En la sección 2.1 se define Prueba de Software y otros términos relacionados. En la sección 2.2 se describen consideraciones importantes a tener en cuenta cuando se esta probando. En la sección 2.3 se presentan los tipos de prueba: unitaria, de integración y del sistema y la distinción entre prueba funcional y estructural. En la sección 2.4 se describen otros tipos como la prueba de regresión, la prueba de humo y la automatización de las pruebas. En la sección 2.5 se presentan las técnicas de prueba agrupadas según sean técnicas de caja negra, caja blanca, basadas en la experiencia o en quién realiza las pruebas. En el caso de las técnicas de caja negra, se brinda una descripción mas detallada, debido a que estas técnicas serán usadas en la definición de ProTest. En la sección 2.6 se introduce en el tema de cómo medir las pruebas. Por último en la sección 2.7 se discute sobre la independencia y el outsourcing de las pruebas.
.

2.1 Definiciones

En esta sección se presentan los principales términos relacionados con las Pruebas de Software. Se definen Prueba de Software, Verificación, Validación y su relación con las Pruebas. Se discuten las distintas definiciones y se precisan las que se eligen para este trabajo. Se utilizará el término Prueba como traducción del término Testing de Software. Se precisan los términos error, defecto y falla. Se exponen los conceptos relacionados con los Requerimientos del Software y su correlación con las Pruebas. Por último, se describe la terminología usada al probar un producto de software.

2.1.1 Pruebas

La primera referencia a las Pruebas *de Software* puede ser rastreada a 1950, pero fue recién en 1957 que la prueba fue distinguida del debugging [Het88]. Dijkstra en 1970 presentaba una importante limitación, que "La Prueba de Software puede ser usada para mostrar la presencia de bugs, pero nunca su ausencia" [Dij70]

Myers en su segunda edición del libro "The art of Software testing" del año 2004, que fué publicado originalmente en 1979, comenta el hecho de que en el tiempo transcurrido entre una edición y otra, la prueba de software no se ha convertido en una ciencia exacta, y que esta lejos de eso. De hecho, parece que se sabe mucho menos sobre la prueba de software, que sobre cualquier otro aspecto relacionado con el desarrollo de software. La prueba continúa estando entre las "artes oscuras" del desarrollo de software [Mye04].

Existen enfoques dinámicos y estáticos para las pruebas. Los enfoques dinámicos apuntan a ejecutar una parte o todo el software para determinar si funciona según lo esperado. El enfoque estático se refiere a la evaluación del software sin ejecutarlo usando mecanismos automatizados (herramientas asistidas) y manuales tales como controles de escritorio, inspecciones y revisiones [Dai94].
En este trabajo se adopta el enfoque dinámico de las Pruebas, tomando como definición de Prueba la de SWEBOK [SWE04]. En SWEBOK se definen Prueba, Prueba de Software, Verificación dinámica y comportamiento esperado de la siguiente manera:

Prueba es una actividad realizada para evaluar la calidad del producto y mejorarla, identificando defectos y problemas.

La **Prueba de software** es la verificación *dinámica* del comportamiento de un programa contra el *comportamiento esperado*, usando un conjunto finito de casos de prueba, seleccionados de manera adecuada desde el dominio infinito de ejecución.

Dinámica: implica que para realizar las pruebas hay que ejecutar el programa para los datos de entrada.

El comportamiento esperado: debe ser posible decidir cuando la salida observada de la ejecución del programa es aceptable o no, de otra forma el esfuerzo de la prueba es inútil. El comportamiento observado puede ser revisado contra las expectativas del usuario, contra una especificación o contra el comportamiento anticipado por requerimientos implícitos o expectativas razonables.

En la literatura existen otras definiciones. Por ejemplo en [IEEE 610.12-1990] de ANSI/IEEE, 1990 se define la Prueba: "Es el proceso de operar un sistema o componente bajo condiciones especificas,

observar o registrar los resultados, y realizar una evaluación de algún aspecto del sistema o componente."

2.1.2 Verificación, Validación y Prueba

Los términos Verificación y Validación suelen usarse indistintamente en algunos contextos. Para las Pruebas del Software ambos términos indican conceptos diferentes:

La definición dada por [IEEE 610.12-1990] de ANSI/IEEE, 1990 de Verificación y Validación es la siguiente:

- **Verificación**: proceso de evaluación de un sistema o componente para determinar si un producto de una determinada fase de desarrollo satisface las condiciones impuestas al inicio de la fase.

- **Validación**: proceso de evaluación de un sistema o componente durante o al final del proceso de desarrollo para determinar cuando se satisfacen los requerimientos especificados.

Boehm [Boe84] usa dos preguntas para clarificar la diferencia entre Verificación y Validación
- Verificación: ¿Estamos elaborando correctamente el producto?
- Validación: ¿Estamos elaborando el producto correcto?

CMMI [CMMI02] define el propósito de la Verificación y la Validación de la siguiente manera:

- **El propósito de la Verificación** es asegurar que los Productos Internos seleccionados cumplen con su especificación de requerimientos. Los métodos de verificación pueden ser, entre otros: inspecciones, revisiones por pares, auditorias, recorridas, análisis, simulaciones, pruebas y demostraciones.

- **El propósito de la Validación** es demostrar que un producto o componente de producto cumple su uso previsto cuando es puesto en su ambiente previsto. Deben ser seleccionados los productos internos (por ejemplo: requerimientos, diseño, prototipos) que mejor indican cuán bien el producto y los productos internos deben satisfacer las necesidades del usuario.

Según Kit [Kit95], la **Verificación** es el proceso de evaluar, revisar, inspeccionar y hacer controles de escritorio de productos intermedios, tales como especificaciones de requerimientos, especificaciones de diseño y código. En el caso del código se refiere a un análisis estático del mismo, y no de su ejecución dinámica. La **Validación** implica ejecutar el código. La Prueba la define como la Verificación más la Validación.

En general en la bibliografía, la Prueba es tomada como una parte del proceso de Verificación y Validación y bajo el entendido de que la Prueba requiere ejecutar el código. En este trabajo se utiliza el enfoque de SWEBOK [SWE04], donde se define que:

La salida observada de la ejecución del programa puede ser comparada mediante:

- **Prueba para la verificación:** El comportamiento observado es revisado contra las especificaciones.
- **Prueba para la validación:** El comportamiento observado es revisado contra las expectativas del usuario

2.1.3 Error, falta, defecto y falla

Se utilizan distintos términos en la bibliografía para distinguir entre la causa de un mal funcionamiento en el sistema, que es percibida por los desarrolladores y la observación de ese mal funcionamiento, que es percibida por quien usa el sistema.

En este trabajo se usará la definición de [Pfl01] cuando se hace referencia a los términos error, falta o defecto y falla:

- **Error (error):** Equivocación realizada por una persona
- **Defecto o Falta (fault):** Es un error al realizar alguna actividad concerniente al software
- **Falla (failure):** Desvío respecto al comportamiento requerido del sistema.

Con estas definiciones, un defecto es una visión interna del sistema, desde la óptica de los desarrolladores, una falla es una visión externa del sistema, o sea un problema que ve el usuario. No todos los defectos corresponden a una falla. Por ejemplo: si un trozo de código defectuoso nunca se ejecuta, el defecto nunca provocará la falla del código.
La Prueba puede revelar fallas, pero son las faltas las que pueden y deben ser removidas [SWE04].

En [IEEE 610.12-1990] de ANSI/IEEE, 1990 se definen:

- equivocación (mistake): Acción del ser humano que produce un resultado incorrecto.
- defecto o falta (fault): Un paso, proceso o definición de dato incorrecto en un programa de computadora. El resultado de una equivocación. (Potencialmente origina una falla).
- falla (failure): Resultado incorrecto, el resultado (manifestación) de una falta.
- error (error): Magnitud por la que el resultado es incorrecto.

En la bibliografía aparece comúnmente la palabra bug para referirse a un defecto o una falta. Un bug se manifiesta como desviaciones del comportamiento previsto del software. Diferentes bugs pueden tener la misma manifestación y un bug puede tener muchos síntomas [Bei90]. Un bug se refiere a cualquier cosa que pueda estar mal en el software. Cuando alguien reporta un bug puede estar haciendo referencia a una falta, una falla o una limitación en el programa que la hace menos valiosa al usuario [KBP01]

2.1.4 Requerimientos y su relación con la Prueba

Los requerimientos son lo que el software debe hacer y/o características que el software debe tener. Deben ser consistentes, completos, realizables y verificables [Bei90].
Los requerimientos del software expresan las necesidades y las restricciones puestas en un producto de software que contribuyen a la solución de un cierto problema del mundo real. Una característica esencial de todos los requerimientos del software es que sean comprobables. Una afirmación es verificable si se puede diseñar un experimento que demuestre o refute la verdad de la sentencia [Bei90].
Los requerimientos funcionales describen las funciones que el software debe realizar. Los requerimientos no funcionales son las restricciones a las posibles soluciones. Éstos últimos, pueden ser clasificados según si son requerimientos de desempeño, requerimientos de seguridad, requerimientos de la confiabilidad, o uno de muchos otros tipos de requerimientos del software.

La *especificación de requerimientos del software* refiere típicamente a la producción de un documento, o a su equivalente electrónico, que puede ser revisado, evaluado, y ser aprobado sistemáticamente.

Los requerimientos se pueden validar para asegurarse de que se han entendido, y es conveniente que el documento de requerimientos conforma los estándares de la organización, y que es entendible, consistente y completo. La **validación de los requerimientos** se refiere al proceso de examinar el

documento de requerimientos para asegurarse de que define el software correcto, es decir, el software que los usuarios esperan. Una característica esencial de un requerimiento de software es que debe ser posible validar que el producto final lo satisface. Una tarea importante es planificar cómo verificar cada requerimiento. En la mayoría de los casos, esto se hace con el diseño de las pruebas de aceptación [SWE04].

2.1.5 Elementos de la Prueba

A continuación se definen los principales conceptos usados para la prueba de un producto de software:

Una **prueba** (test) es:

i. Una actividad en la que un sistema o componente es ejecutado bajo condiciones especificadas , los resultados son observados o registrados y una evaluación es hecha de algún aspecto del sistema o componente.
ii. Un conjunto de uno o más casos de prueba [IEEE 610.12-1990].

Un **caso de prueba** (test case) es un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y poscondiciones de ejecución, desarrollados con un objetivo particular o condición de prueba, tal como ejercitar un camino de un programa particular o para verificar que se cumple un requerimiento especifico [IEEE 610.12-1990].

Un **procedimiento de prueba** (test procedure) es:

i. Instrucciones detalladas para la configuración, ejecución y evaluación de los resultados para un caso de prueba determinado.
ii. Un caso de prueba puede ser usado en más de un procedimiento de prueba [IEEE 610.12-1990].

Un **resultado real** (actual result) es el comportamiento producido u observado cuando un componente o sistema es probado [ISQ05].

Un **resultado esperado** (expected result) es el comportamiento predecido por la especificación u otra fuente, del componente o sistema a ser probado bajo condiciones especificadas [ISQ05].

Un **programa de prueba** (test script) es un programa que especifica para la prueba: elemento a ser probado, requerimiento, estado inicial, entradas, resultados esperados y criterio de validación [Bei90].

Un **conjunto de prueba** (test suite) es un conjunto de una o más pruebas, con un propósito y base de datos común que usualmente se ejecutan en conjunto.

Un **ciclo de prueba** (test cycle) es la ejecución del proceso del testing contra una versión identificada del producto a probar [ISQ05].

Los **datos de prueba** (test data) son los datos que existen (por ejemplo, en una base de datos) antes de que una prueba sea ejecutada, y que afecta o es afectado por el componente o sistema a probar [ISQ05].

La **ejecución de la prueba** (test execution) es el proceso de ejecutar una prueba para el componente o sistema a probar, produciendo el resultado real [ISQ05].

2.2 Consideraciones respecto a la Prueba

El problema fundamental respecto a la Prueba de Software es que no se puede probar completamente un programa, por lo que en el momento de realizar las pruebas se deben tomar decisiones respecto a cómo se van a diseñar los casos de prueba. Otro punto importante a tener en cuenta es la actitud que debe tener la persona que realiza las pruebas.

2.2.1 No es posible probar completamente un programa

Para probar completamente un sistema se deben ejercitar todos los caminos posibles del programa a probar. Myers mostró en 1979 un programa que contenía un loop y unos pocos if, este programa tenia 100 trillones de caminos, un tester podía probarlos todos en un billón de años.
La prueba exhaustiva requiere probar el comportamiento de un programa para todas las combinaciones validas e invalidas de entradas. Además se debe probar esto en cada punto donde se ingresan datos, bajo cada estado posible del programa en ese punto. Esto no es posible, ya que el tiempo requerido es prohibitivo.
En un mundo ideal, desearíamos probar cada permutación posible de un programa. Incluso un programa aparentemente simple puede tener centenares o millares de combinaciones posibles de la entrada y de la salida. Crear los casos de prueba para todas estas posibilidades no es económicamente factible. Este problema fundamental, tendrá implicancias para la economía de las pruebas, suposiciones que el tester tendrá que hacer sobre el programa, y la manera en el cual los casos de prueba se diseñan. El objetivo debe ser maximizar la producción de las pruebas, esto es, maximizar el número de los errores encontrados por un número finito de los casos de prueba [Mye04]. La forma en cómo se seleccionan los casos de prueba es una de las principales decisiones a tomar, teniendo en cuenta las siguientes consideraciones:

Los casos de prueba deben ser revisados regularmente, escribiendo nuevas pruebas que ejerciten diferentes partes del software, con el objetivo de encontrar más defectos [ISQ05].

Otro aspecto a considerar al realizar pruebas es decidir cuándo el programa falla para un conjunto de datos de entrada, o sea, conocer cuál es la salida esperada del programa. Este problema es conocido como el problema del oráculo. Un **oráculo** es cualquier agente (humano o mecánico) que decide si un programa se comportó correctamente en una prueba dada, y produce por consiguiente un veredicto de "pasó" (pass) o de "falló" (fail). Existen diversas clases de oráculos, y la automatización del oráculo puede llegar a ser muy compleja y costosa [SWE04]. El oráculo más común es el oráculo entrada/salida, que especifica la salida esperada para una entrada específica [Bei90]. Si el programa hace exactamente lo que su especificación dice que debe hacer y no hace nada más, entonces cumple con su especificación. No es razonable decir que un programa es correcto si cumple con su especificación, ya que las especificaciones pueden tener errores. [KFN99].

2.2.2 Actitud frente a las pruebas

La Prueba debe ser visto como el proceso destructivo de encontrar errores (cuya presencia se asume) en un programa. Si el propósito del tester es verificar que el programa funciona correctamente, entonces el tester esta fallando al conseguir su propósito cuando encuentra defectos en el programa. En cambio, si el tester piensa que su tarea es encontrar fallas, los buscara con mayor ahínco que si piensa que su tarea es verificar que el programa no las tiene.
Si el objetivo de la prueba es encontrar errores, un caso de prueba exitoso es uno que hace al programa fallar. Por supuesto que eventualmente se puede usar la prueba para establecer algún grado de

confianza de que el programa hace lo que se supone debe hacer y no hace nada que no se suponga que no deba hacer [Mye04].
El tester debe adoptar una actitud destructiva hacia el programa, debe querer que falle, debe esperar que falle y debe concentrarse en encontrar casos de prueba que muestren sus fallas [KFN99].

2.3 Clasificaciones de la Prueba

Existe en la bibliografía distintas formas de clasificar los tipos de prueba existentes. Se usará en esta sección la clasificación dada por Whittaker en [Whi02], donde se propone clasificar la prueba según qué tipo de prueba se esté haciendo: funcional ó estructural y según cómo se realiza la prueba: unitaria, de integración ó del sistema. Se hace énfasis en la prueba funcional, debido a que el proceso definido utiliza ese tipo para probar un producto de software.

2.3.1 Prueba estructural vs. Funcional

La prueba estructural es también llamada de caja blanca o basada en el código. La prueba funcional es conocida también como basado en la especificación o de caja negra.
El objetivo de la prueba estructural es seleccionar los caminos del programa a ejercitar durante las pruebas.

> El objetivo de la **prueba funcional** es validar cuando el comportamiento observado del software probado cumple o no con sus especificaciones. La prueba funcional toma el punto de vista del usuario [Bei90].
>
> En la **prueba funcional** las funciones son probadas ingresando las entradas y examinando las salidas. La estructura interna del programa raramente es considerada [KFN99].

No hay controversias entre el uso de prueba estructural o funcional, ambas son útiles, ambas tienen limitaciones, ambas apuntan a diferentes tipos de defectos. El arte de la prueba en parte está en cómo balancear las pruebas estructurales y funcionales [Bei90].
Para realizar pruebas funcionales, la especificación se analiza para derivar los casos de prueba. Técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto y conjetura de errores son especialmente pertinentes para las pruebas funcionales. Se deben considerar condiciones inválidas e inesperadas de la entrada y tener en cuenta que la definición del resultado esperado es una parte vital de un caso de la prueba. Finalmente, conviene recordar que el propósito de la prueba funcional es mostrar errores y discrepancias con la especificación y no demostrar que el programa cumple su especificación [Mye04].

2.3.2 Prueba unitaria, de integración y del sistema

Según el objeto a probar, las pruebas se clasifican como unitarias, de integración o del sistema. El objeto de la prueba puede variar: un solo módulo, un grupo de módulos (relacionados por el propósito, el uso, el comportamiento o la estructura) o un sistema entero [SWE04].

> La **prueba unitaria** es el proceso de probar los componentes individuales (subprogramas o procedimientos) de un programa. El propósito es descubrir discrepancias entre la especificación de la interfase de los módulos y su comportamiento real [Kit95].

Una unidad es la pieza de software mas pequeña que se puede probar, es en general el trabajo de un programador y consiste de cien o menos líneas de código [Bei90]. La prueba unitaria es conocida también como prueba de módulo o prueba de componente.

> La **prueba de integración** es el proceso en el cual los componentes son agregados para crear componentes más grandes. Es el testing realizado para mostrar que aunque los componentes hayan pasado satisfactoriamente las pruebas unitarias, la combinación de componentes es incorrecta o insatisfactoria [Bei90].

Existen distintas estrategias para combinar los componentes en una integración: para el software estructurado jerárquicamente, se pueden usar una estrategia incremental o una estrategia big bang. Se entiende por desarrollo jerárquico cuando el producto se desarrolla de forma tal que un modulo principal invoca sub-módulos que a su vez invocan módulos de nivel inferior hasta que finalmente se invoca a un modulo que realiza el trabajo.

En la estrategia incremental, cada pieza de software es primero probada por separado mediante prueba unitaria. Luego se prueban las combinaciones de piezas del producto, realizando prueba de integración. Así se van formando grupos integrados que se van probando con otros grupos hasta que todo el sistema es probado junto. Esta estrategia requiere de stubs y drivers para llevarse a cabo.
En la estrategia big bang, la integración entre módulos no es probada hasta que todo el sistema es integrado. En [KFN99] se discuten las ventajas del primer enfoque sobre el segundo.

La prueba incremental a su vez puede ser realizado de dos formas, Bottom up o Top down.
En la estrategia Bottom up, los módulos de nivel mas bajo son probados primero. Se utilizan drivers para invocarlos, luego se integran los módulos del siguiente nivel de jerarquía y se prueban y así sucesivamente.
En la estrategia Top down, los módulos de nivel mas alto son probados primero. Se utilizan stubs para reemplazar los módulos de menor nivel que son invocados, luego se integran los módulos del siguiente nivel de menor jerarquía y se prueban y así sucesivamente [KFN99].

Estrategias de integración más modernas son aquellas guiadas por la arquitectura, que implican integrar los componentes o los subsistemas de software basados en los hilos funcionales identificados [SWE04].

> La **prueba del sistema** refiere al comportamiento del sistema entero. La mayoría de las faltas funcionales deben haber sido identificadas ya durante las pruebas de unidad e integración. La prueba del sistema generalmente se considera apropiada para probar los requerimientos no funcionales del sistema, tales como seguridad, desempeño, exactitud, y confiabilidad. Las interfaces externas, los dispositivos de hardware, o el ambiente de funcionamiento también se evalúan a este nivel [SWE04].

Las pruebas unitarias, de integración o del sistema pueden realizarse desde el punto de vista funcional o estructural, así por ejemplo, podemos hacer prueba unitaria para mostrar que la unidad no satisface su especificación funcional o podemos mostrar que la implementación de la unidad no satisface la estructura de diseño definida.

A continuación se describen los tipos de prueba del sistema [Mye04]:

Prueba de Volumen: Este tipo de prueba del sistema somete el programa a grandes volúmenes de datos. El propósito es demostrar que el programa no puede manejar el volumen de datos especificados en sus objetivos. El volumen a probar puede requerir recursos significativos, en términos del tiempo de procesador y de personas.

Pruebas de Estrés: Estas pruebas someten al programa a cargas pesadas. Una carga pesada es un volumen máximo de datos, o de actividad, en un plazo corto de tiempo. La prueba de estrés implica como elemento el tiempo.

Pruebas de Usabilidad: Intentan encontrar problemas de usabilidad del sistema en su interacción con humanos. El análisis de factores humanos es una cuestión altamente subjetiva.

Pruebas de Seguridad: Intentan subvertir los chequeos de seguridad del sistema. Una forma de pensar tales casos de prueba es estudiar problemas conocidos de seguridad en sistemas similares.

Pruebas de Desempeño: Muchos programas tienen objetivos específicos de desempeño o de eficiencia, indicando características tales como tiempos de respuesta y rendimiento bajo ciertas condiciones de carga de trabajo y configuración. En la prueba de desempeño, los casos de prueba se deben diseñar para demostrar que el programa no satisface sus objetivos de desempeño.

Pruebas de Almacenamiento: Algunos programas tienen establecidos límites en el almacenamiento, que indican, por ejemplo, la cantidad de memoria principal y secundaria que el programa utiliza. Se diseñan los casos de prueba para demostrar que estos objetivos de almacenamiento no se han resuelto.

Pruebas de Configuración: Implican probar el programa con distintas configuraciones de hardware y software posibles
Pruebas de Compatibilidad/ Configuración/ Conversión: La mayoría de los programas que se desarrollan no son totalmente nuevos; son a menudo reemplazos de un sistema existente. Algunas veces, estos programas tienen objetivos específicos referentes a su compatibilidad y procedimientos de conversión del sistema existente. Se diseñan los casos de prueba para demostrar que los objetivos de la compatibilidad no se han resuelto y que los procedimientos de conversión no funcionan.

Pruebas de Instalación: La prueba de instalación es una parte importante del proceso de prueba del sistema, particularmente en un sistema empaquetado con instalación automatizada.

Pruebas de Confiabilidad: Estos tipos de prueba tienen como objetivo la mejora de la confiabilidad del programa, probando que las declaraciones específicas sobre confiabilidad se cumplen.

Pruebas de Recuperación: Algunos programas tienen requerimientos específicos de recuperación que indican cómo el sistema se recupera de errores de programación, de faltas del hardware, y de errores en los datos. Un objetivo de la prueba del sistema es demostrar que estas funciones de recuperación no funcionan correctamente

Pruebas de Mantenimiento: Algunos programas tienen objetivos de mantenimiento, como por ejemplo: requerimientos específicos sobre las ayudas que se proporcionarán al sistema, los procedimientos de mantenimiento, y la calidad de la documentación de la lógica interna.

2.4 Otros tipos de Prueba

Otros tipos de pruebas existentes son las pruebas de regresión, las pruebas de humo o las pruebas automatizadas que se describen a continuación.

2.4.1 Pruebas de Regresión

Su objetivo es verificar que no ocurrió una regresión en la calidad del producto luego de un cambio, asegurándose que los cambios no introducen un comportamiento no deseado u errores adicionales. Implican la reejecución de alguna o todas las pruebas realizadas anteriormente [Bla02].

Después que los testers reportan los defectos, los desarrolladores generan una nueva versión del software, en la cual los defectos supuestamente han sido removidos. La cuestión de fondo es: ¿Cuánta prueba de la versión n es necesario hacer usando las pruebas ejecutadas contra la versión n-1? [Whi00]
Cualquier arreglo de un problema detectado puede [Whi00]:

- Solucionar solo el problema que fue reportado.
- Fallar en solucionar el problema que fue reportado.
- Solucionar el problema pero arruinar otra cosa que antes funcionaba.
- Fallar en solucionar el problema que fue reportado y romper otra cosa que antes funcionaba.

Hay tres tipos de pruebas de regresión [KBP01]:

- Regresión de defectos solucionados: Cuando se reporta un defecto y vuelve una nueva versión que supuestamente lo soluciona, el objetivo es probar que no fue solucionado.
- Regresión de defectos viejos: Se prueba que un cambio en el software causó que un defecto que ya había sido solucionado vuelva a aparecer.
- Regresión de efectos secundarios: Implica volver a probar una parte del producto. El objetivo es probar que el cambio ha causado que algo que funcionaba ya no funciona.

2.4.2 Prueba de Humo

Las pruebas de humo son un conjunto de pruebas aplicadas a cada nueva versión, su objetivo es validar que las funcionalidades básicas de la versión se cumplen según lo especificado. Estas pruebas buscan grandes inestabilidades o elementos clave faltantes o defectuosos, que hacen imposible realizar la prueba como fué planificado para la versión. Si la versión no pasa las pruebas de humo, no se comienza la ejecución de las pruebas planificadas de la versión [KBP01].

2.4.3 Pruebas Automatizadas

Hay herramientas que apoyan diversos aspectos de la prueba. A continuación se presenta una clasificación posible para las herramientas [ISQ05]:

- Administración de las pruebas y el proceso de pruebas: Herramientas para la administración de las pruebas, para el seguimiento de incidentes, para la gestión de la configuración y para la administración de requerimientos.
- Pruebas estáticas: Herramientas para apoyar el proceso de revisión, herramientas para el análisis estático y herramientas de modelado.
- Especificación de las pruebas: Herramientas para el diseño de las pruebas y para la preparación de datos de prueba

- Ejecución de las pruebas: herramientas de ejecución de casos de prueba, herramientas de pruebas unitarias, comparadores, herramientas de medición del cubrimiento, herramientas de seguridad.
- Desempeño y la monitorización: Herramientas de análisis dinámico, herramientas de desempeño, de carga y de estrés, herramientas de monitorización

Comprar o alquilar una herramienta no garantiza el éxito con ella. Cada tipo de herramienta requiere esfuerzo adicional para poder alcanzar ventajas a largo plazo. Entre las ventajas de usar herramientas se encuentran: reducir el trabajo repetitivo (por ejemplo en las pruebas de regresión), mayor consistencia y capacidad de repetición, evaluación objetiva, facilidad para el acceso a la información de las pruebas. Entre los riesgos de usar las herramientas, se encuentran: expectativas poco realistas para la herramienta, subestimar el tiempo, costo y esfuerzo de inducción inicial en la herramienta, subestimar el tiempo y el esfuerzo necesarios para alcanzar ventajas significativas y continuas con la herramienta, subestimar el esfuerzo requerido para mantener los elementos de prueba generados por la herramienta, tener demasiada confianza en la herramienta sustituyendo todas las pruebas manuales [ISQ05].

Para la automatización de las pruebas funcionales son especialmente indicadas las herramientas de captura y reproducción. Estas herramientas permiten al tester capturar y grabar pruebas, para luego editarlas, modificarlas y reproducirlas en distintos entornos. Herramientas que graban la interfase de usuario a nivel de componentes y no de bitmaps son más útiles. Durante la grabación se capturan las acciones realizadas por el tester, creando automáticamente un script en algún lenguaje de alto nivel. Luego el tester modifica el script para crear una prueba reusable y mantenible. Este script se vuelve la línea base y luego es reproducido en una nueva versión, contra la cual es comparado. En general estas herramientas vienen acompañadas de un comparador, que compara automáticamente la salida en el momento de ejecutar el script con la salida grabada [DRP99].

2.5 Técnicas de Prueba

En la literatura existen distintas formas de agrupar las técnicas de Prueba. En este trabajo se agrupan fusionando la clasificación usada en [SWE04] y [KBP01]. En la Tabla 1 se muestran las agrupaciones definidas y las técnicas comprendidas en cada una. A continuación se explica brevemente cada técnica.

Agrupación	Técnicas
Técnicas de Caja Negra	Partición de Equivalencia Análisis del Valor Límite Tablas de decisión Máquinas de estado finito Grafo Causa Efecto Prueba de Casos de Uso Prueba de Dominios
Técnicas de Caja Blanca	Basadas en el Flujo de Control Basadas en el Flujo de los datos Mutantes
Técnicas según quién hace la Prueba	Pruebas de Aceptación Pruebas Alfa y beta Pruebas de usuario Pruebas en pares
Técnicas Basadas en la experiencia	Prueba Ad hoc Conjetura de errores Testing Exploratorio

Tabla 1 – Técnicas de Prueba

2.5.1 Técnicas de Caja negra

Se llama técnicas de caja negra o técnicas de prueba funcional a aquellas donde los casos de prueba se derivan a partir de la especificación del sistema. Las técnicas de caja negra se describen en mayor profundidad que las otras, ya que el proceso de prueba definido en este trabajo es para realizar prueba funcional de productos de software.

2.5.1.1. Partición de Equivalencia

Un buen caso de prueba es aquel que tiene una probabilidad razonable de encontrar un error, y dado que la prueba exhaustiva es imposible, las pruebas se limitan a intentar con un subconjunto pequeño de todas las entradas posibles. Con esta técnica, se intenta seleccionar el subconjunto con la probabilidad más alta de encontrar la mayoría de los errores y que cumpla con las siguientes características:

- Reducir el número de otros casos de prueba que se deban desarrollar. Esto implica que cada caso de prueba debe invocar tantas entradas como sea posible para reducir al mínimo el número total de los casos de prueba necesarios.
- Cubrir otros casos posibles de prueba. Esto implica que se debe intentar repartir el dominio de la entrada del programa en un número finito de clases de equivalencia donde se pueda asumir que probar un valor representativo de cada clase es equivalente a probar cualquier otro valor. Es decir, si un caso de la prueba en una clase de equivalencia detecta un error, se espera que el resto de los casos de prueba en la clase de equivalencia encontraran el mismo error.

La primera consideración se utiliza para desarrollar un sistema mínimo de casos de prueba que cubren estas condiciones. La segunda consideración para desarrollar un sistema de condiciones "interesantes" que se probarán. El diseño de casos de prueba siguiendo partición de equivalencia se divide en dos pasos [Mye04]:

- Identificación de las clases de equivalencia: son identificadas tomando cada condición de la entrada (generalmente una oración o una frase en la especificación) y repartiéndola en dos o más grupos. Dos tipos de clases de equivalencia son identificados: las clases de equivalencia válidas representan entradas válidas al programa, y las clases de equivalencia inválidas representan el resto de los estados posibles (es decir, valores erróneos de la entrada).
- Definición de los casos de prueba: A partir de las clases de equivalencia se identifican los casos de prueba. Se escriben casos de prueba que cubren tantas clases de equivalencia válidas como sea posible y casos de prueba que cubran cada clase de equivalencia inválida. La razón para esto es no enmascarar errores.

2.5.1.2. Análisis del valor límite

La experiencia demuestra que los casos de prueba que exploran condiciones límites tienen una rentabilidad mayor. Las condiciones límite son situaciones en los bordes, por arriba, y por debajo de las clases de equivalencia para los valores de la entrada y de la salida [Mye04].

2.5.1.3. Tablas de decisión

Las tablas de decisión representan relaciones lógicas entre las condiciones (entradas) y las acciones (salidas). Los casos de prueba son derivados sistemáticamente considerando cada combinación posible de condiciones y de acciones [SWE04].
Son una buena manera de capturar los requerimientos del sistema que contienen condiciones lógicas. Pueden ser utilizadas para registrar reglas de negocio complejas de un sistema. Se analiza la

especificación, y las condiciones y las acciones del sistema se identifican. Las condiciones y las acciones de la entrada se indican de una manera tal que puedan ser verdaderas o falsas (booleano). Cada columna de la tabla corresponde a una regla de negocio que define una combinación única de las condiciones que dan lugar a la ejecución de las acciones asociadas a esa regla. La idea es tener por lo menos una prueba por columna, que implica cubrir todas las combinaciones para accionar condiciones.
La fortaleza de las tablas de decisión es crear combinaciones de condiciones que pueden no ejercitarse de otra manera. Puede ser aplicada cuando la acción del software depende de varias decisiones lógicas [ISQ05].

2.5.1.4. Máquinas de estado finito

Modelando un programa como máquina de estado finito, las pruebas se pueden seleccionar para cubrir estados y sus transiciones [SWE04].
Un sistema puede tener distintas respuestas dependiendo de las condiciones actuales o de la historia anterior (su estado). En ese caso, ese aspecto del sistema se puede mostrar como máquinas de estado. Esta forma de modelar permite ver el software en términos de sus estados, las transiciones entre los estados, las entradas o los acontecimientos que disparan el cambio de estado (las transiciones) y las acciones que pueden resultar de esas transiciones. Los estados del sistema son identificables y finitos en número. Las pruebas se diseñan para cubrir una secuencia típica de estados, para cubrir cada estado, para ejercitar cada transición, para ejercitar secuencias específicas de transiciones o para probar transiciones inválidas. La técnica es conveniente para modelar un objeto del negocio que tiene estados específicos o flujos de dialogo de pantalla [ISQ05].

2.5.1.5. Grafo Causa Efecto

Una debilidad del análisis del valor límite y de la partición en clases de equivalencia es que no exploran combinaciones de los valores de la entrada. El grafo Causa Efecto ayuda a seleccionar, de una manera sistemática los casos de prueba. Una causa es una condición de entrada o una clase de equivalencia de las condiciones de la entrada. Un efecto es una condición de salida o una transformación del sistema.

A partir de la especificación, se identifican las causas y los efectos. El contenido semántico de la especificación es analizado y transformado en un grafo booleano que liga las causas con los efectos. Luego, se convierte el grafo en una tabla de decisión donde cada columna en la tabla representa un caso de la prueba.

La técnica de grafo causa-efecto no explora adecuadamente condiciones de límite. El aspecto más difícil de la técnica es la conversión del grafo en la tabla de decisión. Este proceso es algorítmico y se puede automatizar [Mye04].

2.5.1.6. Prueba de Casos de Uso

Un caso de uso es una secuencia de acciones que un sistema realiza con el fin de lograr un resultado de valor para un actor particular. Un actor es alguien o algo fuera del sistema que interactúa con el sistema [JBR99]. Cuenta con precondiciones, que deben cumplirse para que el mismo se realice con éxito y poscondiciones, que son los resultados observables y el estado final del sistema después de que se haya terminado el caso de uso. Un caso de uso tiene un escenario o flujo principal y varios flujos alternativos.
Las pruebas se pueden especificar a partir de casos de uso o escenarios del negocio. Los casos de uso describen los escenarios a través de un sistema basado en su uso probable real, por lo que los casos de prueba derivados a partir de ellos son muy útiles para encontrar defectos en los escenarios durante el uso real del sistema. Los Casos de Uso son muy útiles para diseñar pruebas de aceptación con la

participación del cliente o del usuario y ayudan a descubrir defectos en la integración causados por la interacción de diversos componentes, que la prueba individual del componente no considera [ISQ05].

2.5.1.7. Prueba de Dominios

Un dominio es un conjunto que incluye todos los posibles valores de una variable para una función. En la prueba de dominio se identifican las variables y las funciones. Las variables pueden ser de entrada o de salida. Para cada una, se toman pocos valores representativos de los posibles de la clase de equivalencia (típicamente casos bordes) para cada clase. La hipótesis del método es que si se prueba con unos pocos elementos pero bien representativos de la clase, se encontrarán la mayoría de los defectos que se encontrarían si se prueba con cada miembro de la clase. El elemento primario de interés es la variable más que la función, la prueba de dominio debe analizar una variable y basado en ese análisis, ejecutar pruebas que involucren esta variable en cada función como entrada o salida [KBP01].

2.5.2 Técnicas de Caja Blanca

Se llama técnicas de caja blanca o técnicas de prueba estructural a aquellas donde los casos de prueba se derivan a partir de la estructura del sistema. Requieren conocer el código del programa a probar.

2.5.2.1. Basadas en el flujo de control

Los criterios basados en el flujo de control o la estructura del programa, cubren todas las sentencias o bloques de sentencias en un programa, o combinaciones especificadas de ellas. Se han propuesto varios criterios de cobertura. El más fuerte de los criterios es el de flujo de control, que ejecuta todas las trayectorias del flujo del control de la entrada a la salida. Puesto que la prueba de la trayectoria no es generalmente factible debido a los bucles, otros criterios menos rigurosos son utilizados en la práctica, por ejemplo de cobertura de sentencia, de condiciones y de decisión. La adecuación de tales pruebas se mide en porcentajes; por ejemplo, se dice haber alcanzado una cobertura de sentencia del 100% cuando todos las sentencias han sido ejecutadas por lo menos una vez por las pruebas [SWE04].

2.5.2.2. Basadas en el flujo de los datos

La prueba en el flujo de datos usa la información sobre cómo se definen, se utilizan, y se destruyen las variables del programa. El criterio más fuerte, all definition-use paths, requiere que para cada variable, cada segmento de la trayectoria del flujo del control desde una definición de esa variable hasta su uso sea ejecutado. Para reducir el número de las trayectorias requeridas se utilizan estrategias más débiles como all-definitions y all-uses [SWE04].

2.5.2.3. Mutantes

Un mutante es una versión levemente modificada del programa a probar, diferenciado de él por un cambio sintáctico pequeño. Cada caso de prueba ejercita el programa original y todos los mutantes generados: si un caso de prueba identifica la diferencia entre el programa y un mutante, se dice que el mutante fue "matado".
Puede ser usado para evaluar un conjunto de prueba o como criterio de prueba en sí mismo. En este último caso, las pruebas se diseñan específicamente para matar a mutantes que sobreviven. Para que la técnica sea eficaz, se deben derivar automáticamente de una manera sistemática una gran cantidad de mutantes [SWE04].

2.5.3 Técnicas según quién hace la Prueba

Existen algunas técnicas de prueba que dependen de quién realiza las pruebas, este es el caso de las pruebas de aceptación, las pruebas alfa y beta, las pruebas de usuario y las pruebas en pares.

2.5.3.1. Prueba de Aceptación

La prueba de aceptación es el proceso de comparar el programa contra sus requerimientos iniciales y las necesidades reales de los usuarios. Es realizado generalmente por el cliente o el usuario final. En el caso de un programa por contrato, se realiza la prueba de aceptación comparando la operación del programa contra el contrato original [Mye04].
Generalmente se realiza seleccionando un subconjunto de los casos de prueba del sistema que demuestra formalmente las funcionalidades claves para su aprobación final. Es realizado comúnmente luego que el sistema es instalado en el ambiente del usuario [Dai94].

2.5.3.2. Pruebas Alfa y Beta

Antes de que el software se libere, se distribuye a un pequeño grupo representativo de los usuarios potenciales para el uso interno (alfa) o externo (beta). Estos usuarios reportan problemas con el producto. El uso de las pruebas alfa y beta es a menudo descontrolado, y no siempre es citado en el plan de prueba [SWE04].

2.5.3.3. Prueba de usuario

Es la prueba realizada por el tipo de persona que usará el producto. Puede se realizado en cualquier momento durante el desarrollo, en el lugar del cliente o en el de desarrollo, en ejercicios completamente dirigidos o a la discreción del usuario [KBP01].

2.5.3.4. Prueba en pares

Consiste en que dos testers trabajan juntos para encontrar errores. Comparten una computadora e intercambian el control de la misma mientras prueban [KBP01].

2.5.4 Técnicas basadas en la intuición o en la experiencia

Existen técnicas de prueba que se basan en la intuición o la experiencia de la persona que realiza las pruebas, dentro de estas técnicas se encuentran las pruebas ad hoc, la conjetura de errores y el testing exploratorio.

2.5.4.1. Prueba Ad hoc

Quizás la técnica más practicada sigue siendo la prueba ad hoc: las pruebas son derivadas confiando en la habilidad, intuición, y experiencia con programas similares. La prueba ad hoc puede ser útil para identificar pruebas que no son fácilmente encontradas por técnicas más formales [SWE04].

2.5.4.2. Conjetura de errores

Se ha observado a menudo que ciertas personas sin usar ninguna metodología particular, parecen tener una destreza para encontrar los errores. Dado un programa particular, conjeturan, por la intuición y la

experiencia, ciertos tipos probables de errores y después escriben casos de prueba para exponer esos errores. La idea básica es enumerar una lista de errores y después escribir los casos de prueba basados en la lista. Otra idea es identificar los casos de prueba asociados a asunciones que el programador pudo haber hecho cuando leía la especificación (es decir, las cosas que fueron omitidas de la especificación, por accidente o porque eran obvias) [Mye04].
Los casos de prueba son diseñados pensando las faltas más probables de un programa dado. Una buena fuente de información es la historia de las faltas descubiertas en proyectos anteriores, así como la experiencia de quien prueba [SWE04].

2.5.4.3. Testing Exploratorio

El término "testing exploratorio" fué introducido por Kaner, se trata de ejecutar las pruebas a medida que se piensa en ellas, sin gastar demasiado tiempo en preparar o explicar las pruebas, confiando en los instintos. El testing exploratorio se define como el aprendizaje, el diseño de casos de prueba y la ejecución de las pruebas en forma simultánea. En otras palabras, es cualquier prueba en la cual quien prueba controla activamente el diseño de las pruebas mientras las pruebas son realizadas, y utiliza la información obtenida mientras prueba para diseñar nuevas y mejores pruebas [Bac03]. En el testing exploratorio siempre se debe tomar nota de lo que se hizo y lo que sucedió [KFN99]. Los resultados del testing exploratorio no son necesariamente diferente de aquellos obtenidos de la prueba con diseño previo y ambos enfoques para las pruebas son compatibles [Bac01].
El testing exploratorio puede ser realizado en cualquier situación donde no sea obvio cual es la próxima prueba que se debe realizar. También se realiza cuando se requiere obtener retroalimentación rápida de cierto producto o funcionalidad, se necesita aprender el producto rápidamente, se quiere investigar y aislar un defecto en particular, se quiere investigar el estado de un riesgo particular, o se quiere evaluar la necesidad de diseñar pruebas para esa área. Una estrategia básica para realizar testing exploratorio es tener un plan de ataque general, pero permitirse desviarse de él por periodos cortos de tiempo. Cem Kaner llama a esto el principio del "tour bus", las personas bajan del bus y conocen los alrededores. La clave es no perderse el tour entero [Bac03].
En general, el testing solamente exploratorio puede funcionar para testers con mucha experiencia. Como ventaja se encuentra que es barato y rápido, como contra que no se tienen medidas de cubrimiento y no deja documentación [Bla02]. La técnica " Session Based Test Management " de James Bach [Bac00], es una forma de hacer testing exploratorio que lleva muy poca documentación. Su principal ventaja es que es de bajo costo y medible. Su desventaja es que requiere testers preparados.

2.6 Medir la Prueba

La medición permite crear una memoria organizacional y ayuda a contestar distintas preguntas asociadas con cualquier proceso del software. Mejora la planificación del proyecto, permitiendo determinar las fortalezas y debilidades de los procesos y productos. La medición también ayuda, durante el transcurso de un proyecto, a determinar su progreso, tomar acciones correctivas y evaluar el impacto de tal acción. La medición, para ser efectiva debe: enfocarse en objetivos, aplicarse a todos los productos, procesos y recursos del ciclo de vida, interpretarse basándose en la caracterización y conocimiento de la organización [Bas88].

Las mediciones permiten cuantificar tanto el proceso como el producto. Proporcionan la visión del desempeño del proceso permitiendo: desarrollar perfiles de los datos de los proyectos anteriores que se pueden utilizar para la planificación y mejora del proceso; analizar un proceso para determinar como mejorarlo; determinar la eficacia de modificaciones en el proceso [PMC05].

Una medición es la asignación de números a objetos o eventos de acuerdo con una regla derivada de un modelo o de una teoría. La teoría de una medición debe considerar los siguientes factores [Kan01]:

- El atributo a ser medido: la escala apropiada para el atributo y la variación del atributo
- El instrumento que mide el atributo: la escala del instrumento y la variación de las medidas hechas con ese instrumento
- La relación entre el atributo y el instrumento
- Los efectos secundarios probables de usar ese instrumento para medir ese atributo
- El propósito de tomar la medida
- El alcance de la medida

En el proceso de prueba las mediciones pueden ser usadas para [ISQ05]:

- Monitorizar el proceso de prueba: Mostrar visibilidad sobre las actividades de pruebas. Esta información puede ser usada para medir el criterio de terminación de las pruebas y evaluar el progreso contra lo planificado.
- Reportar las pruebas: Métricas recolectadas al finalizar cada etapa de prueba para evaluar la adecuación de los objetivos de la etapa, la adecuación de la estrategia de pruebas tomada y la efectividad de las pruebas con respecto a sus objetivos.
- Controlar las pruebas: Acciones correctivas tomadas como el resultado de la información y las métricas tomadas y reportadas.

2.7 Outsourcing e Independencia de las Pruebas

El "outsourcing" refiere al proceso en el cual una compañía transfiere todos o una parte de sus departamentos a un proveedor externo que maneja los asuntos de la compañía por un precio definido en el contrato del outsourcing [Kan00].

La motivación de contar con un equipo de prueba independiente del desarrollo surge para mitigar la falta de objetividad al realizar la prueba que puede ocurrir cuando la organización que desarrolla es la misma que escribe las pruebas. Esto, sumado a las presiones institucionales por salir al mercado hace que en muchos casos la calidad del software se conozca recién cuando se pone en producción. La realización de pruebas independientes también implica beneficios desde el punto de vista de los objetivos de las organizaciones involucradas. Para la organización que realiza las pruebas el objetivo es el de encontrar defectos, para la organización que desarrolla el producto el objetivo es construirlo. En organizaciones de pequeño y mediano porte la subcontratación de los servicios de prueba es imprescindible para implementar la prueba independiente, ya que es difícil sustentar la existencia de un área dedicada exclusivamente a las pruebas dentro de la organización misma.

Las ventajas que se obtienen al tener una organización de prueba independiente de las de desarrollo son: motivación en el proceso de pruebas, oposición de intereses con la organización de desarrollo, separación del proceso de pruebas del control gerencial de la organización de desarrollo y el conocimiento especializado que la organización independiente tiene respecto a las pruebas [Mey04].

Cierto grado de independencia es más eficaz en encontrar defectos y faltas. Existen distintos niveles de independencia [ISQ05]:

- Pruebas diseñadas por las personas que escribieron el software (nivel bajo de independencia).
- Pruebas diseñadas por personas distintas pero del equipo de desarrollo.
- Pruebas diseñadas por personas de otro grupo de la organización (por ejemplo: un equipo independiente de prueba).
- Pruebas diseñadas por personas de otra organización o compañía (es decir outsourcing o certificación por un organismo externo).

Dentro de las desventajas se encuentra que la organización de pruebas independiente puede no tener el conocimiento necesario del dominio del negocio, lo que puede llevar a que olvide aspectos importantes

o los subestime. Además, si la curva de aprendizaje del producto es elevada, la transmisión de conocimiento puede ser demasiado costosa y se corre el riesgo de que con el tiempo el equipo de pruebas independiente sea el único que conoce como probar el producto.

Los desarrolladores pueden probar su propio código, pero para ayudar a mantener el foco y obtener una visión independiente esta responsabilidad se asigna a un tester. Esto se debe a que es un recurso entrenado. La eficacia de encontrar defectos puede ser mejorada usando testers independientes, las opciones para la independencia son [ISQ05]:

- Testers independientes dentro del equipo de desarrollo.
- Equipo de pruebas independiente dentro de la organización, reportando a la gerencia de proyecto o la gerencia ejecutiva.
- Especialistas independientes para pruebas especificas tales como usabilidad o seguridad
- Testers independientes externos a la organización.

Capítulo 3

PROCESOS

En este capítulo se describe la relación entre proceso, ciclo de vida del software y proceso de desarrollo. Se estudian los procesos de prueba existentes y su relación con los procesos de desarrollo de software.

El proceso de desarrollo de software describe la vida de un producto de software desde su concepción hasta su entrega, utilización y mantenimiento. Los modelos de mejora de los procesos indican pautas que las organizaciones deben seguir para mejorar sus procesos de desarrollo de software, y con esto, mejorar la calidad del producto que genera.

En forma análoga, el proceso de prueba describe la forma en que el producto de software debe ser probado desde su concepción. Al igual que para el proceso de desarrollo, existen modelos de mejora del proceso de prueba.

También existen modelos que se basan en evaluar para poder mejorar, estos modelos definen mediciones a realizar para poder evaluar el proceso y el producto, y de esa forma mejorar la calidad.

En la sección 3.1 se definen los términos proceso, proceso de desarrollo de software y ciclo de vida, detallando el Modelo V. En la sección 3.2 se presentan modelos de mejora de la calidad y de mejora de las capacidades de los procesos de desarrollo, se resumen el modelo de calidad ISO/IEC 9126, el enfoque Goal-Question-Metric, el modelo de mejora de procesos CMMI y el modelo de proceso personal PSP.

En la sección 3.3 se describen los antecedentes sobre el proceso de prueba, presentando sus participantes y como los distintos autores definen las etapas del mismo. En particular se describen los enfoques de Kit, Black, Kaner, Bach, Prettichord, Hetzel, Whittaker, TMap e ISQTB.

Por último, en la sección 3.4 se describen los modelos de mejora del proceso de prueba Testing Maturity Model (TMM) y Test Process Improvement (TPI).

3.1 Proceso, Ciclo de Vida y Proceso de Software

En [Pfl01] se definen Proceso, Ciclo de Vida y Proceso de Desarrollo de Software de la siguiente manera:

- Un **proceso** es una serie de pasos que involucran actividades, restricciones y recursos que producen una determinada salida esperada.
- Cuando el proceso implica la construcción de algún producto, suele referirse al proceso como **ciclo de vida**.
- El **proceso de desarrollo de software** suele denominarse ciclo de vida del software y describe la vida de un producto de software desde su concepción hasta su entrega, utilización y mantenimiento

La definición de un proceso puede ser un procedimiento, una política o un estándar. Los modelos de ciclo de vida del software sirven como definiciones de alto nivel de las fases que ocurren durante el desarrollo. No tienen como objetivo brindar información detallada, pero sí proveer las principales actividades y sus interdependencias. Ejemplos de modelos de ciclos de vida son el modelo en cascada, el de desarrollo evolutivo, desarrollo iterativo e incremental o modelo espiral [SWE04].

Los procesos pueden ser definidos en diferentes niveles de abstracción, varios elementos de un proceso pueden ser definidos, por ejemplo: actividades, productos (artefactos) y recursos. Los procesos son adaptados a las necesidades locales, por ejemplo, el contexto organizacional, el tamaño del proyecto, los requerimientos legales, las prácticas de la industria y la cultura corporativa [SWE04].

Los modelos de procesos son importantes porque proporcionan un orden en el cual el proyecto debe realizar sus tareas. Existen diferentes modelos de procesos para construir el software. El modelo básico usado en los principios del software fue el "Code & fix" que tiene dos etapas: 1) Escribir un código que resuelva un problema dado y 2) Arreglar los problemas en el código. Se codifica y luego se piensa en los requerimientos, diseño, pruebas, mantenimiento [Boe88]. Este modelo no tiene en cuenta las necesidades del cliente y no responde en forma adecuada a cambios en los requerimientos.

Entre los primeros modelos propuestos está el modelo en cascada creado por Royce en 1970. En este modelo las etapas se representan una tras otra en el tiempo: Requerimientos, Diseño, Implementación, Prueba y Mantenimiento. Para comenzar una etapa se debe haber completado la anterior. El mayor problema con este modelo es que no refleja realmente la forma en que se desarrolla el software, ya que el software se desarrolla con cierto grado de repetición que la cascada no incluye. Actualmente, los modelos más modernos, intentan reflejar este comportamiento dividiendo el desarrollo en fases, realizando el desarrollo en forma iterativa e incremental.

En los modelos de procesos en fases, el sistema se diseña de modo que puede ser entregado en piezas. Esto permite que los usuarios dispongan de ciertas funcionalidades mientras las otras están siendo desarrolladas. En el desarrollo incremental, el sistema, tal como esta especificado en el documento de requerimientos, es dividido en subsistemas de acuerdo con su funcionalidad. Se comienza con un subsistema pequeño y se agrega funcionalidad en cada nueva versión. En el desarrollo iterativo se entrega un sistema completo desde el principio y luego se mejora la funcionalidad de cada subsistema con cada versión [Pfl01]. Entre los procesos más populares que usan este enfoque se encuentran el Proceso Unificado de IBM Racional (RUP) y eXtreme Programming (XP).

A continuación se describe el Modelo V que muestra como se relacionan las actividades de prueba con las de análisis y diseño.

3.1.1 Modelo V

El modelo V fue concebido en 1992 como una serie de directivas que describen los procedimientos, los métodos que se aplicarán, y los requerimientos funcionales para las herramientas utilizadas en los sistemas de software que se desarrollen para las fuerzas armadas federales alemanas [Hir00].
El modelo V es una variación del modelo en cascada que demuestra como se relacionan las actividades de prueba con las de análisis y diseño. En la Figura 1 se puede ver que la punta de la V es la codificación, con el análisis y el diseño a la izquierda y la prueba y el mantenimiento a la derecha. El modelo V sugiere que la prueba unitaria y de integración sea hecha también para verificar el diseño, verificando que todos los aspectos del diseño del programa se han implementado correctamente en el código, la prueba del sistema debe verificar el diseño del sistema, asegurando que todos los aspectos del diseño están correctamente implementados. La prueba de aceptación que es dirigida por el cliente, valida los requerimientos. La vinculación entre la parte izquierda y derecha del modelo V implica que si se encuentran problemas durante la verificación y la validación entonces el lado izquierdo de la V puede ser ejecutado nuevamente para solucionar el problema y mejorar los requerimientos, el diseño y el código antes de retomar las pruebas del lado derecho [Pfl01].

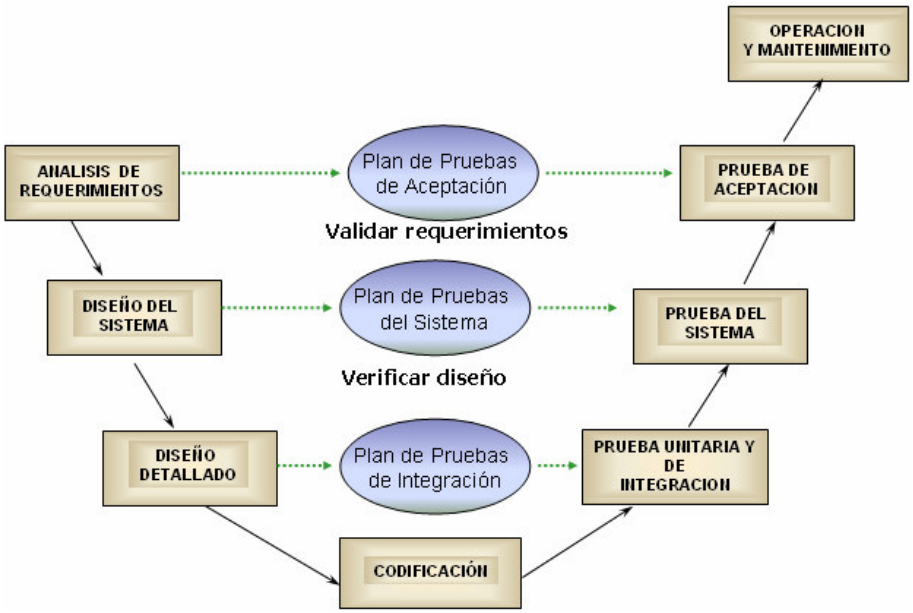

Figura 1 - Modelo V

3.2 Modelos de Calidad y Mejora

En esta sección se presentan modelos vinculados con la calidad y la mejora de los productos y los procesos de software. Se resume el modelo de calidad ISO/IEC 9126, el enfoque Goal-Question-Metric, el modelo de mejora de procesos CMMI y el modelo de proceso personal PSP.

3.2.1 ISO/IEC 9126 – Modelo de Calidad

El modelo de calidad ISO/IEC 9126 [ISO9126] describe un modelo de calidad para un producto de software en dos partes: a) calidad interna y calidad externa, y b) calidad en el uso. La primera parte del modelo especifica seis características para la calidad interna y externa. La segunda parte del modelo especifica cuatro características de calidad en el uso. La calidad en el uso es el efecto combinado para el usuario de las seis características de la calidad del producto de software.

La calidad de proceso contribuye a mejorar la calidad del producto, y la calidad del producto contribuye a mejorar la calidad en el uso. Por lo tanto, evaluar y mejorar el proceso es una forma de mejorar la calidad del producto, y evaluar y mejorar la calidad del producto es una forma de mejorar calidad en uso. De la misma manera, evaluar la calidad en el uso ayuda a mejorar el producto, y la evaluación del producto ayuda a mejorar el proceso, como se muestra en la Figura 2.

Cada característica relevante de la calidad del producto de software debe ser especificada y evaluada, siempre que sea posible, usando una métrica validada o extensamente aceptada. Se definen tres tipos de métricas:

- **Métricas Internas**: Se aplican a un producto de software no ejecutable (tal como código o la especificación) durante el diseño y la codificación. El propósito primario es asegurarse de que la calidad externa requerida y la calidad en uso son alcanzadas.
- **Métricas externas**: Las métricas externas utilizan mediciones de un producto de software derivadas de las mediciones del comportamiento del sistema del cual es parte, mediante la prueba, operación y observación del software o sistema ejecutado.
- **Métricas de calidad en el uso**: Miden el grado en el cual un producto resuelve las necesidades de usuarios especificados de alcanzar los objetivos especificados con eficacia, productividad, seguridad y satisfacción en un contexto especificado de uso.

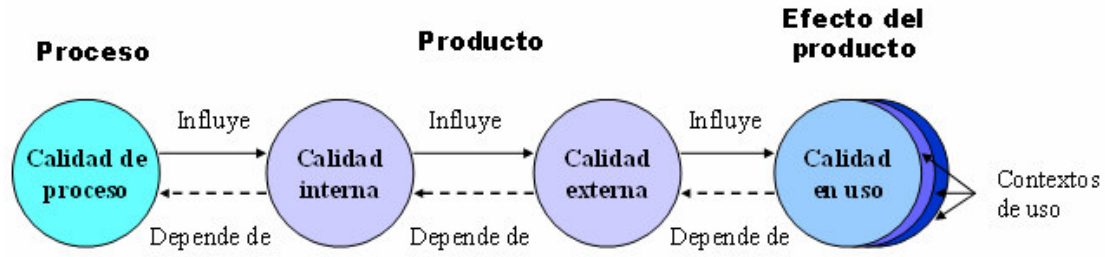

Figura 2- Calidad en el Ciclo de Vida

El **modelo de calidad para la calidad externa e interna** categoriza las cualidades de la calidad del software en seis características: funcionalidad, confiabilidad, utilidad, eficacia, capacidad de mantenimiento y portabilidad, que se subdividen en sub características como se muestra en la Figura 3. Estas sub características se manifiestan externamente cuando el software se utiliza como parte de un sistema informático, y son el resultado de las cualidades internas del software. A continuación se describe cada característica.

- **Funcionalidad:** Capacidad del producto de software para proporcionar las funciones que resuelven las necesidades indicadas e implicadas cuando el software se utiliza bajo condiciones especificadas.
- **Confiabilidad:** Capacidad del producto de software para mantener un nivel especificado de desempeño cuando es utilizado bajo las condiciones especificadas.
- **Usabilidad**: Capacidad del producto de software de ser entendido, aprendido, utilizado y atractivo al usuario, cuando es utilizado bajo condiciones especificadas.

- **Eficiencia**: Capacidad del producto de software para proporcionar el desempeño apropiado, concerniente a la cantidad de recursos usados, bajo condiciones indicadas.
- **Mantenibilidad**: Capacidad del producto de software de ser modificado. Las modificaciones pueden incluir correcciones, mejoras o la adaptación del software a los cambios en el ambiente, los requerimientos y las especificaciones.
- **Portabilidad**: Capacidad del producto de software de ser transferido de un ambiente a otro.

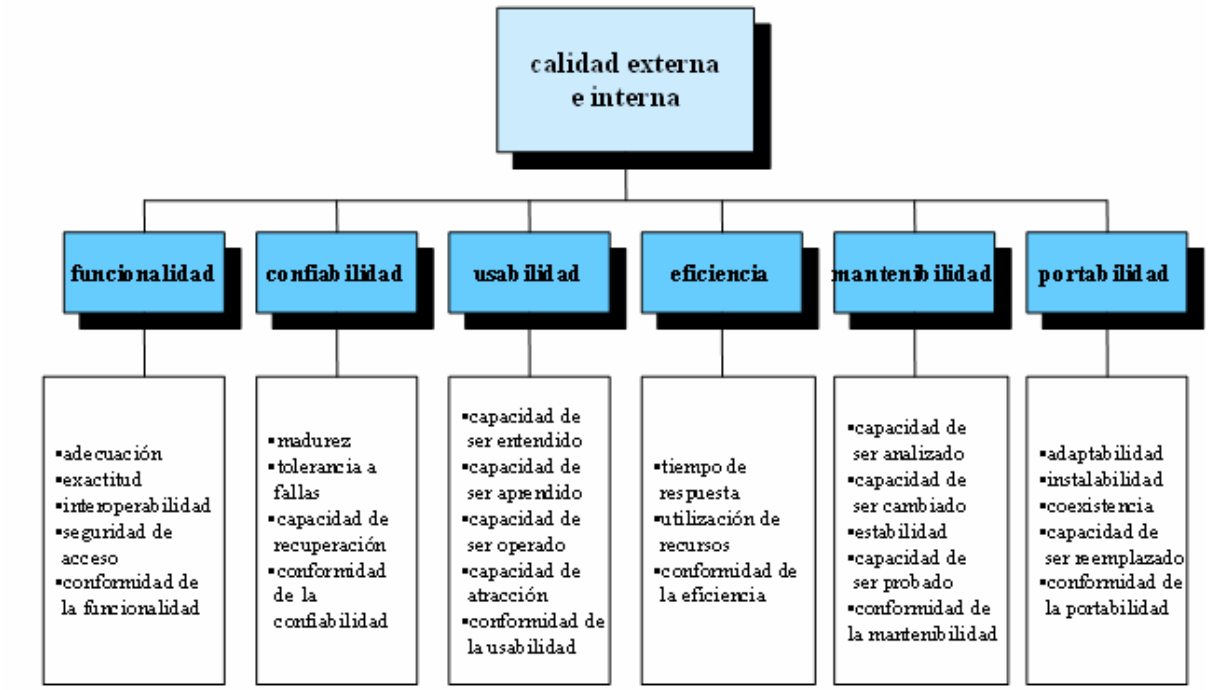

Figura 3 – Modelo de calidad externo e interno

El **Modelo de Calidad en el Uso** tiene cuatro características: eficacia, productividad, seguridad física y satisfacción. La calidad en uso es la visión del usuario de la calidad, es la capacidad del producto de software de permitir a los usuarios alcanzar los objetivos especificados con eficacia, productividad, seguridad y satisfacción en contextos especificados de uso.

- **Eficacia**: Capacidad del producto de software para permitir a los usuarios alcanzar los objetivos especificados con exactitud y completitud, en un contexto especificado de uso.
- **Productividad**: Capacidad del producto de software para permitir a los usuarios utilizar
- cantidades apropiadas de recursos en lo referente a la eficacia alcanzada, en un contexto especificado de uso.
- **Seguridad física**: Capacidad del producto de software para alcanzar niveles aceptables del riesgo de dañar personas, al negocio, al software o al ambiente, en un contexto especificado de uso
- **Satisfacción**: capacidad del producto de software de satisfacer a los usuarios, en un contexto especificado de uso.

En la Tabla 2 se muestran las métricas externas definidas en ISO 9126 que usan métricas del producto de software derivadas de las mediciones del comportamiento del sistema del cual es parte, mediante la prueba, operación y observación del software o sistema ejecutado, permitiendo evaluar calidad del producto de software durante la prueba.

Característica	Nombre	Propósito	Método	Fórmula	Interpretación
Funcionalidad - Adecuación	Adecuación funcional	Cuan adecuadas son las funciones evaluadas?	Nro de funciones que son capaces de realizar las tareas especificadas comparado con el nro de funciones evaluadas	X=1-A/B; A=Nro de funciones con problemas; B= Nro de funciones evaluadas;	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado
	Completitud de la implementación funcional	Cuan completa es la implementación de acuerdo con la especificación de requerimientos?	Las funciones probadas (prueba de caja negra) se comportan de acuerdo con la especificación de requerimientos Contar el numero de funciones faltantes detectadas comparado con el numero de funciones descritas en la especificación de requerimientos	X=1-A/B; A=Nro de funciones faltantes; B= Nro de funciones descritas en la especificación de requerimientos;	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado
	Cubrimiento de la implementación funcional	Cuan correcta es la implementación funcional?	Las funciones probadas (prueba de caja negra) se comportan de acuerdo con la especificación de requerimientos Contar el numero de funciones implementadas incorrectamente o faltantes detectadas comparado con el numero de funciones descritas en la especificación	X=1-A/B; A=Nro de funciones implementadas incorrectamente o faltantes; B= Nro de funciones descritas en la especificación de requerimientos; Cuenta el nro de funciones que están completas contra las que no	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado
	Estabilidad de la especificación funcional	Cuan estable es la especificación funcional previo a entrar en operación?	Contar el numero de funciones descritas en la especificación funcional que han cambiado después de que el sistema es puesto en operación y compararlo con el numero de funciones descritas en la especificación de requerimientos	X=1-A/B; A=Nro de funciones cambiadas después de entrar en operación, comenzando cuando se entro en operación; B= Nro de funciones descritas en la especificación de requerimientos;	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado
Funcionalidad - Precisión	Precisión con las expectativas	Las diferencias entre lo real y lo esperado son aceptables?	Hacer los casos de prueba y comparar la salida con los resultados esperados	Contar el numero de casos de prueba encontrados por los usuarios con una diferencia inaceptable con los resultados esperados	X=A / T; A= Nro de casos con una diferencia contra lo esperado mayor a lo aceptable; T= Tiempo de operación
Confiabilidad - Madurez	Densidad de fallas contra los casos de prueba	Cuantas fallas fueron detectadas durante el periodo de prueba?	Contar el numero de fallas encontradas y el numero de casos de prueba realizados	X=A1/A2; A1= Nro de fallas detectadas; A2= Nro de casos de prueba realizados	0<=X; En etapas finales, cuando mas chico mejor.
	Resolución de las fallas	Cuantas fallas fueron resueltas?	Cuenta el nro de fallas que no reocurrieron durante el periodo de prueba bajo condiciones similares	X= A1 / A2; A1=Nro de fallas resueltas; A2= Nro total de fallas detectadas	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado
	Cubrimiento de las pruebas	Cuanto de los casos de prueba requeridos durante las pruebas han sido ejecutados?	Contar los casos de prueba ejecutados y comparar con los casos de prueba requeridos	X=A/B; A=Nro de casos de prueba ejecutados; B= Nro de casos de prueba a ser ejecutados para cubrir los requerimientos;	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado
	Madurez de las pruebas	Esta el producto bien probado?	Contar el numero de casos de prueba ejecutados que pasaron y compararlos con el numero total de casos de prueba a requeridos	X=A/B; A=Nro de casos de prueba ejecutados que pasaron; B= Nro de casos de prueba a ser ejecutados para cubrir los requerimientos;	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado
Confiabilidad - Tolerancia a fallas	Prevención de caidas	Cuan seguido el producto de software causa la caída de todo el entorno de producción?	Contar el numero de caídas respecto a la cantidad de fallas	X=1-A/B; A=Nro de caídas; B= Nro de fallas	0 <= X <= 1; Cuanto mas cercano a 1.0, más adecuado

Tabla 2 – Métricas de ISO 9126

3.2.2 Enfoque Goal - Question- Metric (GQM)

El enfoque Goal-Question-Metric (GQM) [Bas88] que es traducido en este trabajo como Objetivo-Pregunta-Métrica se basa en la hipótesis de que para que la medición en una organización sea útil se debe primero especificar los objetivos de la misma y sus proyectos, después se deben definir los datos para esos objetivos, y finalmente proporcionar un marco para interpretar los datos con respecto a los objetivos indicados. El resultado del uso de GQM es la especificación de un sistema de medición que toma un conjunto particular de elementos y un conjunto de reglas para la interpretación de los datos medidos.
Un modelo GQM es una estructura jerárquica, como se muestra en la Figura 4, comenzando con un objetivo (que especifica el propósito de la medida, el objeto a ser medido, el elemento a ser medido, y el punto de vista desde el cuál se toma la medida). El objetivo se refina en varias preguntas. Cada pregunta se refina en métricas, algunas objetivas y otras subjetivas. La misma métrica puede usarse para contestar distintas preguntas bajo el mismo objetivo. Varios modelos GQM pueden tener preguntas y métricas en común, teniendo en cuenta al momento de tomar la medida los distintos puntos de vista.
Una vez que se desarrolló un modelo GQM, se selecciona el conjunto de datos, las técnicas, herramientas y procedimientos. Los datos son recolectados e interpretados según el modelo.

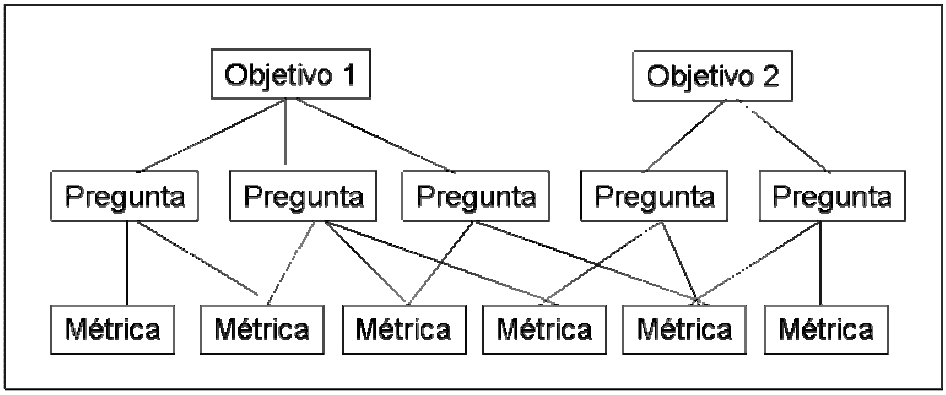

Figura 4 – Esquema GQM

3.2.3 Capability Maturity Model Integration (CMMI)

Capability Maturity Model Integration (CMMI) [CMMI] es la evolución del Capability Maturity Model (CMM), permitiendo la mejora continua en múltiples disciplinas además del software. CMMI ayuda a las organizaciones a mejorar sus procesos de desarrollo de productos y servicios, adquisición y mantenimiento. Existen dos representaciones de CMMI: continua y en etapas.
La representación continua se enfoca en medir la mejora continua del proceso usando niveles de capacidad. Los niveles de capacidad mejoran áreas individuales del proceso como son gestión de la configuración o verificación. La representación en etapas se enfoca en medir la mejora del proceso usando niveles de madurez. Los niveles de madurez mejoran todo el proceso de la Organización en cada nivel. La organización de las áreas de proceso para la representación continua es similar a la definida en ISO/IEC 15504, mientras que la representación en etapas se corresponde con CMM.

Los principales componentes de CMMI en ambas representaciones son:

- Áreas del proceso: Agrupa las prácticas relacionadas a la mejora de esa área.
- Objetivos específicos: Muestra qué debe ser realizado para satisfacer esa área del proceso. Ayudan a determinar si un área del proceso se satisface.

- Objetivos genéricos: Describen la institucionalización que en la organización se debe lograr. Alcanzar los objetivos globales en un área del proceso significa que se mejoró la planificación e implementación del proceso asociado con esa área.

Las áreas del proceso son agrupadas en cuatro categorías: Gerenciamiento del Proceso, Gerenciamiento del Proyecto, Ingeniería y Soporte. Dentro de la categoría de Ingeniería se cubren las actividades de desarrollo y mantenimiento de software: Requerimientos, Administración de Requerimientos, Solución Técnica, Integración del Producto, Verificación, Validación.

El **área del proceso de Verificación** asegura que los productos internos cumplen con sus requerimientos especificados, mientras que el **área de proceso de Validación** valida incrementalmente los productos contra las necesidades del usuario.
La Validación demuestra que el producto cumple su uso previsto mientras que la verificación evalúa cuando un producto interno refleja los requerimientos especificados. La verificación asegura que "se está construyendo bien" mientras que la validación asegura que "se está construyendo lo correcto".
A continuación se describen las áreas de proceso de Verificación y de Validación según el CMMI en su representación continua. En su correspondiente representación en etapas, las áreas de proceso de Verificación y Validación pertenecen al nivel de madurez 3, donde los procesos para cada área están bien caracterizados y entendidos y son descritos en estándares, procedimientos, herramientas y métodos.
En CMMI se usa el término Producto Interno (Work Product), para referirse a cualquier artefacto producido por un proceso.

Verificación

El propósito de la Verificación es asegurar que los Productos Internos seleccionados cumplen con su especificación de requerimientos. El área de proceso Verificación en CMMI incluye la selección, inspección, pruebas, análisis y demostración de productos internos. Los métodos de verificación pueden ser, entre otros: inspecciones, revisiones por pares, auditorias, recorridas, análisis, simulaciones, pruebas y demostraciones.

Para lograr los Objetivos Específicos de esta área, las prácticas de Verificación a realizarse se dividen de la siguiente forma:

1. Preparación: Incluye seleccionar los Productos Internos para la Verificación, establecer el entorno de Verificación, establecer los procedimientos y criterios de Verificación
2. Realizar Revisiones por Pares: Las revisiones por pares implican una examinación metódica de los productos internos por un par para identificar defectos y recomendaciones de cambios necesarios.
3. Verificar los productos internos seleccionados: Incluye realizar la verificación de los productos internos seleccionados contra sus requerimientos, registrar el resultado de las actividades de verificación, identificar acciones resultantes de la verificación de los productos internos y documentar el método de verificación real utilizado y las desviaciones de los métodos y procedimientos definidos. Los resultados son analizados y se identifican acciones correctivas

Ejemplos de mediciones usadas para controlar el proceso de Verificación contra lo planificado, objetivo genérico del nivel de capacidad 2, incluyen:

- Perfil de Verificación: Número de verificaciones planificadas versus realizadas, defectos encontrados.
- Número de defectos encontrados por categoría de defectos.
- Reportes de tendencias en los problemas encontrados, por ejemplo cantidad de problemas abiertos y cerrados.
- Reportes de estado de los problemas encontrados, por ejemplo: cuanto tiempo lleva abierto cada problema reportado.

Validación

El propósito de la Validación es demostrar que un producto o componente de producto cumple su uso previsto cuando es puesto en su ambiente previsto. Deben ser seleccionados los productos internos (por ejemplo: requerimientos, diseño, prototipos) que mejor indican cuán bien el producto y los productos internos deben satisfacer las necesidades del usuario.

Para lograr los Objetivos Específicos de esta área, las prácticas de Validación a realizarse se dividen de la siguiente forma:

1. Preparación: Incluye seleccionar los productos para la Validación y establecer y mantener el ambiente, los procedimientos y los criterios para la Validación.
2. Validar los productos: Los productos son validados para asegurar que son adecuados para su uso en su ambiente operativo. Se realizan las actividades de validación y los resultados son recolectados de acuerdo con los métodos, procedimientos y criterios establecidos. Las desviaciones a los procedimientos durante la ejecución deben ser anotadas. Los datos resultantes de las pruebas de validación, demostraciones o evaluaciones son analizados contra los criterios de validación definidos.

Ejemplos de mediciones usadas para controlar el proceso de Validación contra lo planificado, objetivo genérico del nivel de capacidad 2, incluyen:

- Número de actividades de validación completadas versus planificadas;
- Reportes de tendencia de problemas encontrados (por ejemplo: cantidad encontrados versus cantidad cerrados);
- Reportes de tiempo de los problemas encontrados (por ejemplo: cuánto tiempo ha estado abierto un problema reportado).

3.2.4 Personal Software Process (PSP)

El cuerpo de conocimiento del Personal Software Process (PSP) [PMC05] se estructura en forma jerárquica, los conceptos y las habilidades se describen y se descomponen en tres niveles de abstracción. El término concepto se utiliza para describir la información, los hechos, la terminología, y los componentes filosóficos de la tecnología. El término habilidad refiere a la capacidad de un ingeniero de aplicar los conceptos al realizar una tarea. Los conceptos claves y las habilidades claves forman un Área de Conocimiento, y las áreas relacionadas del conocimiento se clasifican juntas como Áreas de Competencia, como se muestra en la Figura 5.

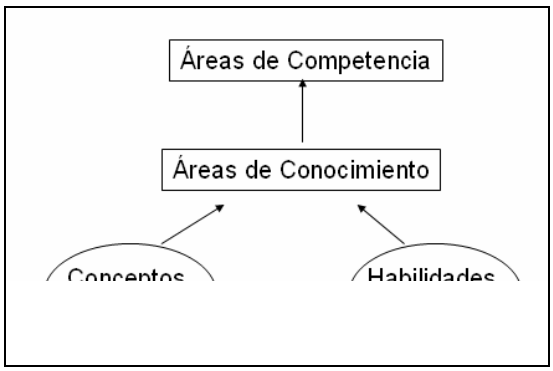

Figura 5 - Estructura jerárquica del Cuerpo de Conocimiento del PSP

- **Área de Competencia**: Amplio grupo de conceptos y de habilidades que un ingeniero está bien calificado para realizar intelectual o físicamente.

- **Área del conocimiento**: La suma de la comprensión específica y de la capacidad ganada a partir de la experiencia o estudio en el área de competencia específica.
- **Concepto clave**: Un principio explicativo aplicable a un caso o a una ocurrencia específica dentro de una área de conocimiento particular.
- **Habilidad clave**: Habilidad, facilidad, o destreza que se adquiere o se desarrolla con el entrenamiento o la experiencia dentro de un Área de Conocimiento particular.

El cuerpo de conocimiento de PSP se compone de siete áreas de competencia. A continuación se describen brevemente cada una de las áreas de competencia.

1. **Conocimiento Fundacional:** Esta área de competencia se ocupa de los conceptos básicos de la mejora de proceso, de los métodos estadísticos en los cuales se construye el PSP y de los elementos genéricos requeridos en cualquier esfuerzo de mejora de proceso.

2. **Conceptos Básicos:** El conocimiento y las habilidades descritas en esta área de competencia son los bloques fundacionales que forman los principios fundamentales en los cuales se construye el PSP.

3. **Medición y Estimación de Tamaño:** Esta área de la capacidad se ocupa de la medición del tamaño y los conceptos de estimación en los cuales se construye el PSP. Los elementos esenciales son la capacidad de definir medidas apropiadas del tamaño y de utilizar métodos disciplinados y datos históricos para estimar tamaño.

4. **Realización y seguimiento de los planes del proyecto:** Esta área de la capacidad se ocupa de la capacidad de utilizar una estimación del tamaño del software para planificar y seguir un proyecto de software. Las partes esenciales de la planificación de un proyecto son la capacidad de construir una agenda, definir las tareas, realizar las tareas conforme la agenda y seguir las tareas respecto al plan.

5. **Planificación y Seguimiento de la Calidad:** Esta área de competencia se ocupa de la necesidad de producir productos que satisfacen las necesidades del usuario, de medir el grado en el cual las necesidades del usuario fueron logradas y de generar productos de calidad.

6. **Diseño del software:** Esta área de competencia se ocupa de la habilidad de incorporar el diseño y las verificaciones de diseño en un proceso PSP.

7. **Extensiones y personalización del proceso:** Esta área de competencia se ocupa de las modificaciones que es necesario realizar al PSP cuando se escala desde programas más pequeños a más grandes, cuando se trabaja con situaciones o ambientes poco familiares o cuando se pasa a un desarrollo basado en el equipo además del trabajo individual.

3.3 Proceso de Prueba

En esta sección se expone el conocimiento que tiene que ver con el proceso de prueba. Se presentan los participantes en el proceso y las distintas fases que se definen en la bibliografía para el proceso de prueba de software.

3.3.1 Participantes

Buscar faltas en un sistema requiere curiosidad, pesimismo profesional, ojo crítico, atención al detalle, buena comunicación con los pares de desarrollo, y experiencia en la cual basarse para conjeturar errores. En [ISQ05] se definen dos roles para las pruebas: líder y tester.
El líder y el tester necesitan buenas habilidades interpersonales para comunicar la información sobre los defectos, el progreso y los riesgos, de una manera constructiva.
El líder de la prueba es quien planifica, monitoriza y controla las actividades y las tareas de las pruebas. Las tareas típicas del líder de la prueba son:

- Coordinar la estrategia de prueba y planificar con los gerentes del proyecto y otros.
- Escribir o revisar la estrategia de prueba para el proyecto, y la política de prueba de la organización.
- Contribuir con la perspectiva de prueba a otras actividades del proyecto, tales como planificar la integración.
- Planificar las pruebas, considerando el contexto y los riesgos. Incluye seleccionar el enfoque de las pruebas, estimar el tiempo, esfuerzo y el costo de la prueba, adquirir los recursos, definiendo niveles de prueba, ciclos, y objetivos, y planificar el seguimiento de incidentes.
- Iniciar la especificación, preparación, implementación y ejecución de las pruebas y monitorizar y controlar la ejecución.
- Adaptar la planificación basada en los resultados y el progreso de las pruebas (documentados en informes de estado) y tomar cualquier acción necesaria para compensar los problemas.
- Instalar una adecuada administración de la configuración para la trazabilidad del testware.
- Introducir métricas convenientes para medir el progreso y evaluar la calidad de las pruebas y el producto.
- Decidir qué se debe automatizar, en qué grado, y cómo.
- Seleccionar herramientas para apoyar la prueba y organizar cualquier entrenamiento necesario en el uso de la herramienta para los testers.
- Decidir sobre la puesta en práctica del ambiente de prueba.
- Agendar las pruebas.
- Escribir los informes basados en la información recopilada durante la prueba.

Dependiendo del nivel de la prueba y de los riesgos relacionados con el producto y el proyecto, diversa gente puede asumir el papel del tester, guardando un cierto grado de independencia. Típicamente los testers en el nivel de componente e integración son los desarrolladores, en el nivel de la prueba de aceptación serían expertos del negocio, y para la prueba de aceptación operacional serían usuarios. Las tareas típicas del tester son:

- Revisar y contribuir a los planes de prueba.
- Analizar, revisar y evaluar los requerimientos del usuario, las especificaciones y los modelos para las pruebas.
- Crear las especificaciones de prueba.
- Instalar el ambiente de prueba.
- Elaborar y adquirir los datos de prueba.
- Implementar las pruebas en todos los niveles, ejecutar y registrar los resultados, evaluar los resultados y documentar las desviaciones de los resultados esperados.
- Usar según se requiera, herramientas para la administración o monitorización de las pruebas.
- Automatizar las pruebas (puede ser ayudado por un desarrollador o un experto en automatización de las pruebas).
- Revisar las pruebas desarrolladas por otros.

3.3.2 Fases de las Pruebas

En esta sección se describen los principales aportes para la elaboración del Proceso de Prueba que se propone en el Capítulo 4. Los distintos autores definen un ciclo de vida específico para las Pruebas, que, en general, separa en distintas etapas las actividades de planificación, diseño y las de ejecución de las pruebas.
Edward Kit [Kit95] define las etapas del proceso de prueba y sus principales actividades. Rex Black [Bla02] se focaliza en la planificación de las pruebas, definiendo las fases sin entrar en detalle en sus actividades. Cem Kaner, James Bach y Bret Pettichord [KBP01] recopilan 293 lecciones aprendidas sobre las pruebas, sin organizarlas en un proceso. Hetzel [Het88] define las fases de prueba acompañando el ciclo de vida del software. Cem Kaner, Jack Falk y Hung Quoc Nguyen [KFN99] describen las técnicas de prueba que son útiles en cada etapa del ciclo de vida del software, sin agruparlas en fases o definiendo actividades específicas. James A. Whittaker [Whi00] y International Software Testing Qualifications Board (ISTQB) [ISQ05] definen las fases del proceso de prueba, sin entrar en el detalle de las actividades que las componen. TMap [PTV02] es el proceso de prueba definido por la empresa Sogeti, define las fases y el detalle de las actividades de prueba.

Edward Kit

Kit [Kit95] divide las actividades de prueba en las siguientes fases: Planificación, Diseño de la Arquitectura de las pruebas, Desarrollo del Testware, Ejecución de las pruebas, Evaluación de las pruebas y Mantenimiento del Testware. A continuación se detalla cada una de estas fases:

Planificación de las pruebas: Se realiza un plan global para todas las actividades de prueba (unitario, integración, usabilidad, funcional, del sistema y de aceptación). El objetivo del plan de pruebas es definir el alcance, enfoque, recursos y agenda de las actividades de prueba.
Diseño de la arquitectura de las pruebas: Es el proceso de diseñar una estructura para las pruebas. Se define cómo se organizan, se categorizan y se estructuran las pruebas y el repositorio de pruebas. Los principales puntos a definir incluyen: Organización de las pruebas respecto a su fuente (requerimientos, funciones, etc.), categorización de las pruebas y convenciones para agruparlas, estructura y convenciones de nombres para el repositorio de pruebas, agrupación de las pruebas según períodos de ejecución.
Desarrollo del Testware: El testware es diseñado para cumplir con los objetivos de detectar tantos errores como sea posible y minimizar el costo de desarrollar las pruebas, de ejecutarlas y de mantenerlas. Las tareas para desarrollar el testware son:

- Diseño detallado: Es el proceso de especificar los detalles del enfoque de las pruebas para los ítems de prueba e identificar los casos de prueba asociados. Los pasos para realizar esta tarea incluyen: identificar los ítems que deben ser probados, asignar prioridades a esos ítems basándose en los riesgos, desarrollar el diseño de las pruebas de alto nivel para grupos de pruebas relacionados, desarrollar casos de pruebas individuales a partir del diseño de alto nivel.
- Implementación: Es el proceso de llevar cada especificación a un caso de prueba pronto para ser ejecutado. Los entregables de salida son: Casos de prueba, datos de prueba, Especificaciones de los procedimientos de prueba y Matriz de cubrimiento.

Ejecución de las pruebas: Proceso de ejecutar todos los casos de prueba seleccionados y observar los resultados. La ejecución de las pruebas incluye las siguientes tareas: Selección de los casos de prueba, configuración, ejecución, análisis posterior, registro de actividades, resultados e incidentes, determinar si las fallas fueron causadas por errores en el producto o en las pruebas, medir el cubrimiento lógico interno. Los principales entregables de la ejecución son: test logs, reportes de incidentes y reporte de cubrimiento lógico.

Evaluación de las pruebas: La evaluación incluye:

- Evaluación del cubrimiento de las pruebas: Se evalúan los casos de prueba en su conjunto y se decide si es necesario o no realizar más pruebas, se estudia el cubrimiento en varios niveles: funciones, requerimientos, lógica.
- Evaluación de los errores del producto: Evaluar la calidad del producto respecto a la ejecución de las pruebas realizadas.
- Evaluación de la efectividad de las pruebas: Evaluar las pruebas respecto al criterio de completitud y decidir cuando parar o agregar más pruebas y seguir, se basa en los dos anteriores.

Rex Black

Black [Bla02] divide el esfuerzo de prueba en las siguientes fases: Planificación, Configuración, Desarrollo de las pruebas y Ejecución de las pruebas, las cuales se describen a continuación:

Planificación: En esta etapa se realiza la planificación del proyecto de prueba, incluye las actividades de definición de riesgos, estudio del cronograma y del presupuesto, escribir el plan de pruebas y seleccionar las herramientas para las pruebas.
Configuración: Incluye obtener el hardware y otros recursos necesarios e instalación del laboratorio de pruebas.
Desarrollo de las Pruebas: Instalar las herramientas de prueba y de reportes, crear las suites de prueba y las librerías de casos de prueba y documentar como es el proceso de prueba que pone los elementos del testware en acción.
Ejecución de las Pruebas: Ejecutar las pruebas, registrar su estado y reportar resultados.

Kaner, Bach y Prettichord

Kaner, Bach y Prettichord [KBP01] definen que la prueba involucra al menos las siguientes cuatro actividades:

Configurar: Preparar el producto para las pruebas. Ponerlo en el estado de comienzo.
Operar: Ingresar los datos al producto y darle los comandos. Interactuar con él de alguna manera estipulada. De otra forma, no se realiza prueba sino revisión.
Observar: Recolectar la información acerca de cómo el producto se comporta, datos de salida, el estado del sistema, interacción con otros productos, etc.
Evaluar: Aplicar reglas, razonamiento o mecanismos que podrían detectar bugs en los datos que se observan.

Bill Hetzel

Hetzel [Het88] divide el esfuerzo de prueba en las siguientes fases, que ocurren en paralelo con el desarrollo:

- Análisis: Se planifica y definen los requerimientos y objetivos de las pruebas.
- Diseño: Se especifican las pruebas a ser desarrolladas.
- Implementación: Se construyen los procedimientos y los casos de prueba.
- Ejecución: Se ejecutan y reejecutan las pruebas.
- Mantenimiento: Se guardan y modifican las pruebas a medida que el software cambia.

La integración de la prueba con el ciclo de vida del software se describe a continuación:

- **Comienzo del Proyecto**: Desarrollar la estrategia de pruebas, establecer el enfoque y esfuerzo de pruebas.
- **Requerimientos**: Establecer los requerimientos para las pruebas, asignar responsabilidades para probar, diseñar los procedimientos de prueba preliminares y las pruebas basadas en requerimientos, validar los requerimientos.
- **Diseño**: Preparar un plan del sistema preliminar, completar el plan de pruebas de aceptación y la especificación del diseño, completar las pruebas basadas en el diseño, validar el diseño.
- **Desarrollo**: Completar el plan de pruebas del sistema, finalizar los procedimientos de prueba y pruebas basadas en el código, completar el diseño de pruebas unitarias o de módulos, probar los programas, integrar y probar los subsistemas, realizar las pruebas del sistema.
- **Implantación**: Realizar las pruebas de aceptación, probar los cambios y arreglos, evaluar la efectividad de las pruebas.

James Whittaker

Whittaker [Whi00] define 4 fases de las pruebas, que permiten a los testers una estructura para agrupar problemas relacionados que deben resolver antes de seguir con la fase siguiente. Las fases definidas son:

Fase 1 - Modelar el ambiente del software: Los testers deben identificar y simular las interfaces que un sistema de software usa y enumerar las entradas que cruzan cada interfase. Los testers deben seleccionar valores para cada variable de entrada y decidir como secuenciar las entradas.
Fase 2 - Seleccionar los escenarios de prueba: Muchos modelos de dominio y partición de variables representan un número infinito de escenarios de prueba, cada uno cuesta tiempo y dinero. Se debe aplicar un criterio de adecuación de los datos de prueba, que debe ser el representante adecuado y económico de todos los posibles test. En general el más adecuado es aquel que encuentra mayor cantidad de defectos.
Fase 3 - Ejecutar y evaluar los escenarios de prueba: Teniendo identificado el conjunto de pruebas, los testers deben ejecutarlos y evaluar los resultados. La ejecución de las pruebas puede ser manual o automatizada. La evaluación de los escenarios involucra la comparación de la salida real, resultante de la ejecución del escenario con su salida esperada como fue documentada en su especificación. La especificación se asume correcta, las desviaciones son fallas. Los testers trabajan junto con los desarrolladores para priorizar y minimizar las pruebas de regresión.
Fase 4 - Medir el progreso de la prueba: Las mediciones no ayudan a saber el progreso de la prueba. Es difícil saber cuando parar de probar, cuando el producto esta pronto para liberarse. Los testers necesitan una medida cuantitativa del número de defectos que quedan en el software y de la probabilidad de que cualquiera de estos bugs sean descubiertos en producción.

Test Management Approach (TMap)

TMap fue creado por la empresa Sogeti [PTV02]. TMap es el enfoque para definir un proceso de prueba estructurado, basado en 4 componentes:

- **Ciclo de vida** que describe las actividades de test que deben realizarse, consistentes con el ciclo de vida del software.
- **Actividades Organizacionales**, que tienen dos componentes:
  1) la organización del equipo de test, donde cada uno debe tener tareas y responsabilidades
  2) la incorporación del equipo de test en la organización del proyecto
- **Infraestructura y herramientas**. El entorno de pruebas debe ser estable, controlable y representativo.

- **Técnicas aplicables al proceso de prueba.** Las técnicas permiten la ejecución de las actividades en forma estructurada y repetible.

Para lograr un proceso estructurado, las distintas componentes deben estar balanceadas, el ciclo de vida es central al resto de las componentes. Cada fase del ciclo de vida de las pruebas requiere una organización específica, infraestructura y técnicas.
El ciclo de vida del TMap consiste de las siguientes fases: Planificación y Control, Preparación, Especificación, Ejecución, Finalización.

- **Planificación y Control:** El objetivo de esta fase es crear un plan de pruebas, que defina quién, cuándo, dónde y cómo se realizan las actividades. Las actividades tienen como objetivo la coordinación, monitorización y control del proceso de prueba y poder conocer la calidad del objeto de las pruebas. El cliente recibe información sobre la calidad del producto y los riesgos mediante reportes periódicos.
- **Preparación:** Determinar si las especificaciones del software escritas o no, tienen la calidad necesaria para la especificación y ejecución de las pruebas.
- Especificación: Incluye la especificación de los casos de prueba y configurar la infraestructura.
- **Ejecución:** Se ejecutan las pruebas especificadas de forma de obtener conocimiento de la calidad del producto a probar.
- **Finalización:** Consiste de conservar los materiales del test para su futura reutilización, realización del informe final y evaluación del proceso de prueba para mejorar el control de futuros procesos de prueba.

International Software Testing Qualifications Board (ISTQB)

El programa del curso para la certificación del tester en el nivel inicial de International Software Testing Qualifications Board (ISTQB) [ISQ05] define el proceso de prueba consistente de las siguientes actividades principales, que si bien son secuenciales, pueden ocurrir concurrentemente.

Planificación y control: La planificación de las pruebas es la actividad donde se definen los objetivos la prueba y las actividades de prueba para resolver esos objetivos. El control de las pruebas es la actividad de comparar el progreso real contra el planificado y reportar el estado, incluyendo desviaciones al plan. Implica tomar las acciones necesarias para lograr los objetivos. La planificación de las pruebas tiene en cuenta la retroalimentación de las actividades de control y tiene las siguientes tareas principales:

- determinar el alcance y los riesgos, identificando los objetivos de las pruebas
- determinar el enfoque de prueba (técnicas, elementos a probar, cobertura, identificando e interconectando los equipos implicados en la prueba, testware)
- determinar los recursos requeridos para las pruebas (personas, ambiente de prueba, PCs)
- implementar la política de prueba y/o estrategia de prueba
- agendar actividades de análisis y diseño de las pruebas
- agendar la implementación, ejecución y evaluación de las pruebas
- determinar los criterios de terminación

El control de las pruebas tiene las siguientes tareas principales:

- medir y analizar los resultados
- supervisar y documentar el progreso, cobertura de las pruebas y criterios de terminación
- tomar acciones correctivas
- tomar decisiones

Análisis y Diseño de las pruebas: Es la actividad donde los objetivos generales de las pruebas se transforman en condiciones de prueba tangibles y diseños de pruebas. El análisis y el diseño de las pruebas tiene las siguientes tareas principales:

- revisar la base de las pruebas (tal como requerimientos, arquitectura, diseño, interfaces)
- identificar las condiciones de prueba o requerimientos de prueba y los datos de prueba requeridos basados en el análisis de los elementos a probar, de la especificación, del comportamiento y de la estructura
- diseñar las pruebas
- evaluación de la testability de los requerimientos y el sistema
- diseñar el ambiente de inicialización de la prueba e identificar cualquier infraestructura requerida y herramientas

Implementación y ejecución: Es la actividad donde las condiciones de prueba se transforman en casos de prueba y testware y se configura el ambiente. La implementación y ejecución de las pruebas tiene las siguientes tareas principales:

- Desarrollar y priorizar casos de prueba, crear datos de prueba, escribir procedimientos de prueba y, opcionalmente: preparación de arneses de prueba y escribir pruebas automatizadas
- Crear conjuntos de prueba (test suite) a partir de los casos de prueba para su ejecución eficiente
- Verificar que se haya instalado el ambiente de la prueba correctamente
- Ejecutar las pruebas manualmente o usando las herramientas de ejecución de pruebas, según la secuencia prevista
- Registrar el resultado de la ejecución de las pruebas y registrar la versión del software probada, las herramientas de prueba y testware
- Comparar resultados reales con resultados esperados
- Reportar discrepancias como incidentes y analizarlos para establecer su causa (por ejemplo: un defecto en el código, en los datos de prueba, en el documento de prueba, o un error en la manera que la prueba fue ejecutada)
- Repetir las actividades de la prueba como resultado de la acción tomada para cada discrepancia. Por ejemplo, reejecutar una prueba que falló previamente para confirmar un arreglo, ejecución de una prueba corregida y/o la ejecución de pruebas para asegurarse de que no fueron introducidos defectos en las áreas sin cambios o de que el arreglo del defecto no trajo otros defectos (prueba de regresión)

Evaluar el criterio de terminación y reportes: Es la actividad donde la ejecución de las pruebas es evaluada contra los objetivos definidos. Esto se debe hacer para cada nivel de la prueba. La evaluación del criterio de terminación tiene las siguientes tareas principales:

- Comparar los registros de las pruebas contra los criterios de terminación especificados en el plan de prueba
- Determinar si son necesarias más pruebas o si los criterios de terminación especificados deben ser cambiados
- Escribir un informe para el cliente

Actividades de cierre: Las actividades de cierre recogen los datos de las actividades terminadas para consolidar experiencia, testware, hechos y números. Las actividades de cierre incluyen las siguientes tareas principales:

- Comprobar que los entregables fueron entregados según lo planificado, el cierre de los reportes de los incidentes que siguen estando abiertos, y la documentación de la aceptación del sistema

- Archivar el testware, el ambiente de la prueba y la infraestructura de prueba para su posterior reutilización
- Analizar lecciones aprendidas para futuras versiones y proyectos, y mejorar la madurez de las pruebas

3.4 Modelos de Mejora de las Pruebas

Se describen a continuación modelos que mejoran el proceso de prueba. Se presenta el modelo de Testing Maturity Model (TMM) y el modelo Test Process Improvement (TPI).

Testing Maturity Model (TMM)

Testing Maturity Model (TMM) [BS196] [BS296] es una guía para las organizaciones que quieren mejorar y evaluar su proceso de prueba. Es un complemento al CMM y existe una correspondencia directa entre los niveles en ambos modelos. TMM contiene un conjunto de niveles para que la organización mejore la madurez de su proceso de prueba, un conjunto de prácticas recomendadas en cada nivel de madurez y un modelo de evaluación que permite a la organización evaluar y mejorar su proceso de prueba. Cada nivel de madurez representa una etapa en la evolución hacia un proceso de prueba maduro. Moverse de un nivel al siguiente implica que las prácticas del nivel anterior continúan realizándose. A continuación se describen las características de cada nivel de TMM, junto con los objetivos de madurez para cada uno.

Nivel 1 – Inicial: El proceso de prueba es caótico. No hay objetivos de madurez planteados para este nivel.

Nivel 2 – Definido: La prueba está separado del debugging y es definido como una fase que sigue a la codificación. Es una actividad planificada. El objetivo primario de la prueba en este nivel es mostrar que el software cumple con sus especificaciones. Los objetivos de madurez para este nivel son:

- Desarrollar objetivos para la prueba y el Debugging.
- Iniciar el Proceso de Planificación de las Pruebas: Planificar las pruebas incluye establecer objetivos, analizar riesgos, delinear estrategias, realizar el diseño de las pruebas y los casos de prueba. Un plan de pruebas debe ser desarrollado y distribuido a los desarrolladores, los requerimientos de usuario deben ser la entrada para el plan de pruebas.
- Institucionalizar las Técnicas y Métodos básicos de Prueba: Debe especificarse qué técnicas y métodos de Prueba deben ser usados en cada caso y qué herramienta lo soporta.

Nivel 3 – Integración: La prueba es parte del ciclo de vida del software. La planificación de la prueba comienza en la fase de Requerimientos y continúa a través del ciclo de vida, siguiendo una variación del Modelo V. Los objetivos de la prueba son establecidos con respecto a los requerimientos basados en las necesidades de cliente y usuarios y son usados para el diseño de las pruebas y los criterios de éxito. Los objetivos de madurez para este nivel son:

- Establecer una Organización de Pruebas: Se identifica a un grupo que es responsable por las pruebas, con roles y responsabilidades definidas. Son responsables por la Planificación de las pruebas, la ejecución y registro, estándares, métricas, base de prueba, reutilización, seguimiento y evaluación de las pruebas.
- Establecer un Programa de Capacitación: Los testers deben estar capacitados de forma que puedan realizar su trabajo de forma eficiente y efectiva
- Integrar la Prueba en el Ciclo de Vida: Se desarrolla y adapta una versión del Modelo V.

Nivel 4 – Gerenciado y Medido: El proceso de prueba es medido y cuantificado. Las revisiones en todas las fases del proceso de desarrollo son reconocidas como actividades de prueba y control de calidad. Los objetivos de madurez para este nivel son:

- Establecer un Programa de Revisión: Las revisiones son realizadas en todas las fases del ciclo de vida para identificar, catalogar y borrar defectos, probando los productos internos temprano y en forma efectiva
- Establecer un Programa de Medición: Permite evaluar la calidad y efectividad del proceso de prueba y la productividad del personal de prueba. Se incluyen las mediciones relacionadas con el progreso de las pruebas, los costos, datos sobre errores y defectos.
- Evaluar la Calidad del Software: Incluye definir atributos de calidad que puedan medirse y objetivos de calidad para evaluar los productos y productos intermedios.

Nivel 5 – Optimización, Prevención de Defectos y Control de calidad: Existen mecanismos para la mejora continua de la Prueba. Se realiza prevención de defectos y control de calidad. Los objetivos de madurez para este nivel son:

- Prevenir los Defectos: Las organizaciones maduras son capaces de aprender de su historia. Los defectos se registran, se analizan los patrones en los defectos y se identifican las causas de los errores.
- Controlar la Calidad: Se utilizan técnicas de muestreo estadístico y niveles de confianza para mejorar el proceso.
- Optimizar el Proceso de Prueba: El proceso de prueba es mejorado continuamente en el proyecto y la organización. Incluye identificar las prácticas de prueba que requieren ser mejoradas, implementar las mejoras, seguir el progreso, evaluación continua de nuevas herramientas y tecnologías a adoptar y apoyo en la transferencia de tecnología.

Test Process Improvement (TPI)

El modelo Test Process Improvement (TPI) [KoP98] brinda guías prácticas para evaluar el nivel de madurez de la prueba en una organización y para mejorar paso a paso el proceso. Los componentes del modelo son:

- **Áreas Claves**: Permiten mejorar y estructurar el proceso de prueba. Existen 20 áreas claves.
- **Niveles**: La forma en que las áreas claves son organizadas en el proceso es lo que determina la madurez del proceso. Existen los niveles: A, B, C y D. Cada nivel mejora el anterior en términos de tiempo, dinero y calidad. Cada nivel consiste de ciertos requerimientos para las áreas claves, los niveles mayores comprenden los requerimientos de los niveles inferiores.
- **Checkpoints**: Los requerimientos para cada nivel están dados como checkpoints.
- **Matriz de Madurez del Test**: No todas las áreas claves son igualmente importantes, los niveles y las áreas claves están relacionadas en una matriz de madurez del test, mostrando las prioridades y dependencias entre los niveles y las áreas claves.

A continuación se describen los componentes y su interacción.

Áreas Claves: Las 20 áreas claves definidas en el TPI son:

- **Estrategia de Test:** La estrategia define que requerimientos y riesgos de calidad son cubiertos por qué pruebas.
- **Modelo de Ciclo de Vida:** Las fases del proceso pueden ser: Planificación, Preparación, Especificación, Ejecución y Finalización. En cada fase se realizan distintas actividades. Para cada actividad deben definirse: propósito, entrada, proceso, salida, dependencias, técnicas aplicables, herramientas, facilidades requeridas, documentación.
- **Momento de participación:** El proceso de test debe comenzar cuanto antes en el ciclo de vida del software, esto ayuda a encontrar defectos y prevenir errores.

- **Estimación y Planificación:** Indican que actividades van a ser realizadas, cuando y cuales son los recursos necesarios.
- **Especificación de Técnicas de Test:** Aplicar una forma estandarizada para derivar las pruebas, la aplicación de técnicas permite conocer la calidad y profundidad de las pruebas e incrementar su reutilización.
- **Técnicas estáticas de Test:** Inspección de programas o evaluación de mediciones son parte de las técnicas estáticas.
- **Métricas:** Métricas sobre el progreso del proceso y la calidad del sistema son importantes para evaluar las consecuencias de ciertas acciones de mejora, comparando datos antes y después de realizar las acciones.
- **Automatización de las pruebas:** La automatización del proceso de prueba tiene como objetivos reducir las horas, mayor profundidad en las prueba, incrementar la flexibilidad de las pruebas, mayor motivación de los testers.
- **Entorno de pruebas:** El entorno debe ser configurado para que los resultados de las pruebas puedan ser determinados. Se deben definir responsabilidades, administración, disponibilidad.
- **Entorno de oficina:** La organización de la oficina tiene influencia positiva en la motivación del personal y la eficiencia del trabajo.
- **Compromiso y motivación:** El compromiso y la motivación de las personas son pre requisitos para ejecutar el proceso de prueba sin problemas.
- **Capacitación:** En distintas disciplinas además de la prueba.
- **Alcance de la metodología:** La metodología usada debe ser lo suficientemente genérica a ser aplicable en toda situación pero contiene suficiente detalle como para no tener que pensar las cosas cada vez.
- **Comunicación:** Dentro del equipo de pruebas y con el resto de las partes como son desarrolladores, cliente, usuarios, etc.
- **Reportes:** Reportar al cliente sobre el producto y el proceso de desarrollo.
- **Administración de los defectos:** Administrar el ciclo de vida de los defectos y realizar un análisis de calidad de la tendencia de los defectos encontrados.
- **Administración del Testware:** Los productos de las pruebas deben ser mantenibles y reusables y además se deben administrar como elementos de configuración.
- **Administración del Proceso de Prueba:** La administración del proceso es de vital importancia para la realización de pruebas buenas.
- **Evaluación:** Implica la inspección de productos intermedios tales como requerimientos y diseño. Los defectos son encontrados en etapas tempranas del proceso de desarrollo.
- **Prueba de bajo nivel:** Incluye pruebas unitarias y de integración. Son realizados por los desarrolladores.

Cada área clave del proceso se evalúa en un nivel para conocer el estado del proceso de prueba, cada nivel tiene requerimientos que se deben cumplir para el área clave. Estos requerimientos se expresan como checkpoints, son preguntas que deben ser contestadas afirmativamente para poder clasificar en cierto nivel. Luego de definir en qué nivel se encuentra una determinada área, se debe conocer qué pasos realizar para mejorar. Los niveles y áreas claves están relacionados en la Matriz de Madurez del Test, muestra las prioridades y dependencias entre los niveles y las áreas claves. Las columnas de la matriz muestran las áreas claves, las filas muestran los niveles de madurez y el elemento de la celda es el nivel que se debe alcanzar en cada área para llegar a ese nivel de madurez.

Capítulo 4

PROTEST

En este capítulo se presenta ProTest, proceso para pruebas funcionales de productos de software realizadas por una organización dedicada a la prueba independiente. ProTest es el proceso seguido en el Laboratorio de Testing Funcional del Centro de Ensayos de Software. Es una guía para realizar prueba funcional a partir de las especificaciones del producto. Entre sus principales características se encuentra que es independiente del ciclo de vida utilizado para desarrollar el producto, basado en el análisis de riesgo del producto para definir las funcionalidades a probar y la prioridad con que serán probadas y organiza el proyecto de prueba en ciclos, donde cada ciclo se corresponde con una versión del producto a probar.

ProTest consta de cuatro etapas y puede ser usado para realizar prueba funcional independiente de productos desde su comienzo, en sucesivos ciclos de desarrollo o en productos que se encuentran en mantenimiento.

A su vez, ProTest define un conjunto de roles responsables de realizar las actividades del proceso y generar sus artefactos.

Este capítulo se organiza de la siguiente manera: en la sección 4.1 se describen las principales características de ProTest, en la sección 4.2 se muestran sus etapas y las actividades en cada etapa, en la sección 4.3 se definen los roles involucrados en ProTest. Por último, en la sección 4.4 se muestra la relación entre ProTest y el ciclo de vida de desarrollo del producto.

4.1 Principales características

En esta sección se presentan las principales características de ProTest. El proceso de prueba definido es independiente del proceso seguido para desarrollar el producto, tiene como objetivo probar las funcionalidades del producto a partir de sus especificaciones, se basa en el análisis de riesgo del producto para definir las funcionalidades a probar y la prioridad con que serán probadas, organiza el proyecto de prueba en ciclos de prueba, que guían el proceso de prueba y acompañan el ciclo de vida del producto.

4.1.1 Prueba independiente

ProTest se utiliza en proyectos de prueba de productos que ocurren en paralelo al proyecto de desarrollo del producto. Es independiente de la metodología seguida para desarrollar el producto a probar. Ha sido concebido como proceso a seguir por una organización que realiza evaluación independiente de productos de software como lo es el Centro de Ensayos de Software.
El proyecto de prueba puede comenzar a la vez que el proyecto de desarrollo del producto, cuando el proyecto de desarrollo del producto está en una etapa más avanzada o cuando el producto ya se encuentra instalado en el ambiente de producción y se está realizando mantenimiento del mismo. En la sección 4.7 se describe como el proyecto de prueba interactúa con el ciclo de vida de desarrollo en cada uno de esos escenarios.

4.1.2 Prueba funcional

El objetivo de la prueba funcional es validar cuando el comportamiento observado del software probado cumple o no con sus especificaciones. Para esto es necesario contar con las especificaciones del producto y una versión ejecutable del mismo. Se revisan las especificaciones para asegurarse de que se pueden definir los casos de prueba a partir de ellas, en el caso de que no existan especificaciones o que las mismas estén incompletas, se trabaja junto al cliente y los desarrolladores para generarlas o mejorarlas.

4.1.3 Prueba basada en los riesgos del producto

ProTest se basa en el análisis de riesgo del producto para definir la prioridad con que se van a definir y ejecutar los casos de prueba.
Como se planteó en el Capítulo 2, la prueba exhaustiva no es posible, por lo que para cada producto, se debe realizar la prueba más efectiva y menos costosa que asegure que el producto es suficientemente confiable, seguro y cumple con los requerimientos de los usuarios. Dado que el tiempo para realizar la prueba nunca es suficiente, deben definirse prioridades. Una estrategia posible es tomar la decisión de qué probar basándose en los riesgos.
Un riesgo es la probabilidad de que algo no deseado ocurra. En la práctica, se le pregunta a los desarrolladores sobre las cosas que le preocupan respecto al producto, para encontrar las funcionalidades a tener especial atención durante las pruebas. Se identifican las partes del sistema que en caso de fallar tienen las consecuencias más serias y aquellas que tienen mayor frecuencia de uso, ya que si una parte del sistema es usada frecuentemente y tiene un error, el uso frecuente hace que se tengan grandes posibilidades de que la falla aparezca. [Kit95]
Según James Bach [Bac99], el enfoque basado en los riesgos tiene tres pasos:

1. Confeccionar una lista priorizada de riesgos,
2. Realizar pruebas que exploran cada riesgo,
3. Cuando un riesgo se mitiga y emergen nuevos, se debe ajustar el esfuerzo de la prueba.

La magnitud de un riesgo es proporcional a la probabilidad y el impacto del problema. Cuanto más probable es que el problema suceda, y más alto el impacto, más alto es el riesgo asociado a ese problema. Existen dos enfoques heurísticos para el análisis de riesgo, uno "desde adentro hacia fuera" y otro "desde afuera hacia adentro". Son acercamientos complementarios, cada uno con sus fortalezas. El enfoque de adentro hacia fuera pregunta "¿qué riesgos se asocian a esta funcionalidad?", mientras que el enfoque de afuera hacia adentro es el opuesto "¿qué funcionalidades se asocian a esta clase de riesgo?".
El enfoque "desde afuera hacia adentro" estudia una lista de riesgos potenciales y se realiza la correspondencia con los detalles del producto. Con este enfoque, se consulta una lista predefinida de riesgos y se determina si aplican [Bac99].
ProTest usa el enfoque "desde adentro hacia fuera" para realizar el estudio de riesgos del producto, el cual se describe a continuación:

Enfoque Desde adentro hacia fuera

Se estudia en detalle el producto y se identifican los riesgos asociados a ellos.
En cada parte del producto, se realizan tres preguntas, buscando:

- Vulnerabilidades: ¿qué debilidades o faltas posibles hay en este componente?
- Amenazas: ¿qué entradas o situaciones hay que pudieran explotar una vulnerabilidad y accionar una falta en este componente?
- Víctimas: ¿quién o qué puede ser impactado por las fallas potenciales y cuán malas pueden ser?

Este enfoque requiere conocimiento técnico, pero no necesariamente de quien prueba. Se le pide al desarrollador en reuniones de trabajo que busque los riesgos, y se discute cómo diseñar las pruebas para evaluar y manejar ese riesgo. Una sesión como esta dura alrededor de una hora, hace que quien prueba entienda la funcionalidad y obtenga una lista de riesgos específicos y de estrategias asociadas de la prueba. Este enfoque requiere habilidades de comunicación, y voluntad de cooperación.
El análisis de riesgo es un método heurístico, por lo que se podrían estar dejando de lado funcionalidades importantes para el negocio. Para evitar que esto ocurra, ProTest realiza además de las pruebas basadas en las especificaciones, testing exploratorio, esta técnica puede ser consultada en la sección 2.5.4.3 del Capítulo 2. En caso de que en una sesión de testing exploratorio se encuentre un defecto crítico para el negocio que no fue encontrado por la prueba planificada para el ciclo, se debe revisar el análisis de riesgo realizado.
Además de realizar análisis de riesgo del producto, ProTest también realiza análisis de riesgo del proyecto. Los riesgos del proyecto son aquellos que hacen peligrar el logro de sus objetivos. Ejemplos de riesgos posibles en un proyecto de prueba son: no detectar los defectos importantes, problemas de comunicación con el equipo de desarrollo, datos difíciles de generar o confidencialidad de los datos, atrasos en la agenda de desarrollo, personal sin entrenamiento, dominio del negocio desconocido, entre otros.

4.1.4 Proceso guiado por Ciclos de Prueba

En un ciclo de prueba se puede ejecutar una, alguna o todas las pruebas planificadas para el producto. Cada ciclo de prueba está asociado a una versión del producto a probar, cada nuevo ciclo de prueba implica una nueva versión de uno o más componentes del sistema [Bla02].
Durante el ciclo de vida de un producto, sin importar cual sea el proceso de desarrollo, se van generando distintas versiones de la aplicación. Las actividades de la prueba se realizan para una determinada versión del producto, sobre la cual se ejecutan las pruebas y se reportan los incidentes encontrados. Las pruebas que serán ejecutadas sobre una versión son planificadas con anticipación y deberían ser ejecutadas, a menos que las prioridades cambien. Es común que mientras se realizan las pruebas para una versión, en paralelo, se estén realizando mejoras a esa versión por parte de los

desarrolladores. Cuando se reporta un incidente, ocurre a menudo que los desarrolladores conocen el motivo de la falla y rápidamente lo arreglan, generando una nueva versión del producto. A menos que realmente no tenga ningún valor seguir probando la versión anterior, el equipo de prueba debería negarse a aceptar esa nueva versión si no estaba planificado, ya que esa nueva versión puede corregir el defecto introduciendo nuevos en funcionalidades que ya fueron probadas, con lo cual, se deberían ejecutar todas las pruebas nuevamente, sin terminar de conocer la calidad global de la versión que se estaba probando. Con cada nueva versión del producto se realizan alguna o todas las tareas asociadas a las pruebas, a esto se le llama un ciclo de prueba.
Uno de los principales desafíos desde el punto de vista de la prueba independiente es estimar cuantos ciclos de prueba se requieren, ya que, en general, no todas las versiones que genera desarrollo llegan a ser probadas por el equipo de prueba, entre dos ciclos de prueba podrían existir más de dos versiones del producto generadas por el equipo de desarrollo.

4.2 Etapas

Las etapas de ProTest son el resultado de fusionar las distintas propuestas discutidas en la sección 2.3.2, Fases de las Pruebas del Capítulo 2. Estas propuestas fueron adaptadas para ser usadas en una evaluación independiente. A diferencia de las otras propuestas, los ciclos de prueba en ProTest son lo que guían el proceso de prueba.

En la Figura 6 se muestran las etapas de ProTest en el tiempo. El proceso cuenta de cuatro etapas: Estudio Preliminar, Planificación, Ciclo de Prueba y Evaluación del Proyecto.

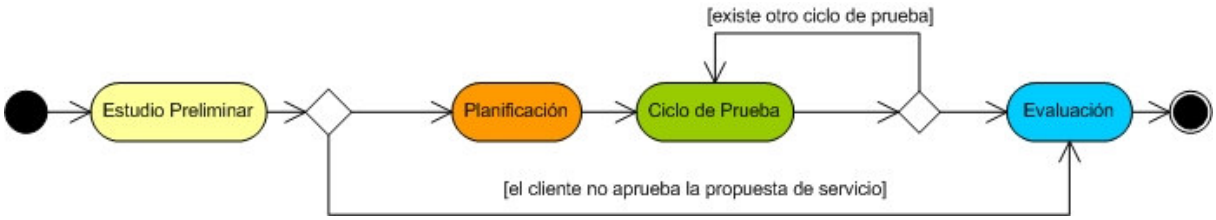

Figura 6 - Etapas de ProTest

Los objetivos de cada una de las etapas definidas son:

Estudio Preliminar: Se estudian las principales funcionalidades del producto, con el objetivo de definir el alcance de las pruebas y un primer cronograma de los ciclos de prueba. A partir de estos datos se realiza la propuesta de servicio de prueba (la forma en que se realiza la cotización no forma parte de ProTest). Si el cliente aprueba la propuesta de servicio de prueba, se sigue con la etapa de Planificación, en caso contrario se pasa a la etapa de Evaluación, donde se analiza cuales fueron los problemas en este proyecto y se archiva para su consulta en futuros proyectos.

Planificación: El objetivo de esta etapa es planificar el proyecto de prueba, una vez que fué aprobado por el cliente. Se definen los ciclos de prueba y las funcionalidades a probar en cada ciclo en función del análisis de riesgo del producto. Se genera el Plan de Pruebas que resume toda la información del proyecto de prueba y las decisiones tomadas durante la etapa de Planificación.

Ciclo de Prueba: El objetivo de esta etapa es generar y ejecutar las pruebas para una versión determinada del producto. El proyecto de prueba es guiado por los ciclos de prueba, cada ciclo de

prueba está asociado a una versión del producto a probar. Esta etapa se divide a su vez en cuatro sub-etapas como se muestra en la Figura 9. Las sub-etapas se describen en la sección 4.3.1 – Ciclos de Prueba.
Evaluación: Esta etapa tiene como objetivo conocer el grado de satisfacción del cliente, realizar el informe final, evaluar el proceso de prueba para su mejora y almacenar los elementos del proyecto de prueba para su uso en proyectos posteriores.

4.2.1 Estudio Preliminar

Esta etapa estudia la factibilidad de realizar el proyecto de prueba. En la Tabla 3 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Actividades	Artefactos generados
I1 – Definición del Servicio	RR – Resumen de Reunión PS – Propuesta de Servicio
I2 - Revisión preliminar de requerimientos	IP – Inventario de Prueba
I3 - Análisis preliminar de riesgo	IP – Inventario de Prueba
G1 – Estimación de las tareas	ET – Estimación de Tareas
G2 – Reportar el Esfuerzo	PE – Planilla de Esfuerzo

Tabla 3 – Estudio Preliminar

Las actividades G1- Estimación de las Tareas y G2 – Reportar el Esfuerzo, son actividades realizadas en todas las etapas del proceso, al comienzo de cada etapa se debe estimar el tiempo que lleva realizar cada una de las actividades de la etapa y luego que las actividades son realizadas, se debe reportar el tiempo insumido en las mismas. Estas actividades generales no se muestran en los diagramas de actividad de las etapas.
En la Figura 7 se muestra el diagrama de las actividades involucradas en esta etapa. La actividad I1- Definición del Servicio, tiene como objetivo llegar a un acuerdo con el Cliente respecto al alcance del proyecto de prueba sobre el cual poder hacer una cotización. En la actividad I2 – Revisión Preliminar de Requerimientos, se estudian las fuentes de requerimientos existentes para saber si es posible derivar las pruebas a partir de ellas y se listan los grupos de funcionalidades del producto, los cuales son analizados y priorizados junto al cliente y desarrolladores en la actividad I3 – Análisis Preliminar de Riesgo del Producto. Con las funcionalidades priorizadas se retoma la actividad I1 – Definición del Servicio para definir el alcance y la agenda del proyecto de prueba. A partir del alcance y la agenda definidos el equipo de prueba realiza la cotización del proyecto de prueba, el procedimiento de cómo realizar la cotización del proyecto de prueba no es parte de ProTest. En la Figura 6 se aprecia que si la cotización es aprobada por el cliente, se sigue en la etapa de Planificación, en caso contrario se realiza la etapa de Evaluación.

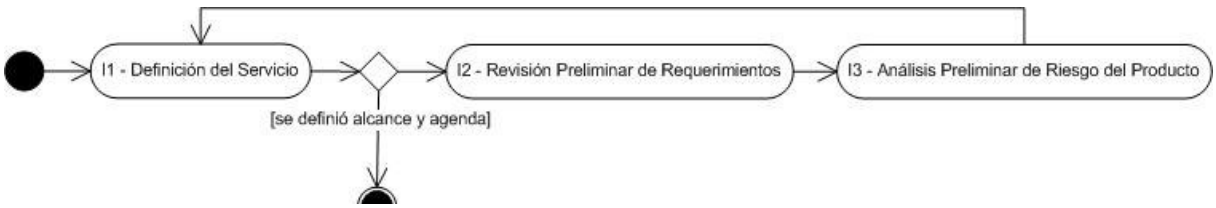

Figura 7 - Actividades en Estudio Preliminar

4.2.2 Planificación

El objetivo de esta etapa es planificar el proyecto de prueba, una vez que el cliente aprobó la cotización del mismo. En la Tabla 4 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Actividades	Artefactos generados
P1 - Negociación con el Cliente	RR – Resumen de Reunión
P2 - Revisión de requerimientos	IP – Inventario de Pruebas
P3 - Análisis de riesgo	IP – Inventario de Pruebas
P4 – Exploración del Producto	IP – Inventario de Pruebas
P5 – Definición de los Ciclos de Prueba	ACP – Agenda de Ciclos de Prueba
P6 – Definición del Testware	TW – Testware
P7 – Planificación de las Pruebas	PP – Plan de Pruebas
P8 – Definición del Proceso de Incidentes	PI – Proceso de Incidentes
G1 – Estimación de las tareas	ET – Estimación de Tareas
G2 – Reportar el Esfuerzo	PE – Planilla de Esfuerzo

Tabla 4- Planificación

En la Figura 8 se muestra el diagrama de las actividades involucradas en la etapa de Planificación.

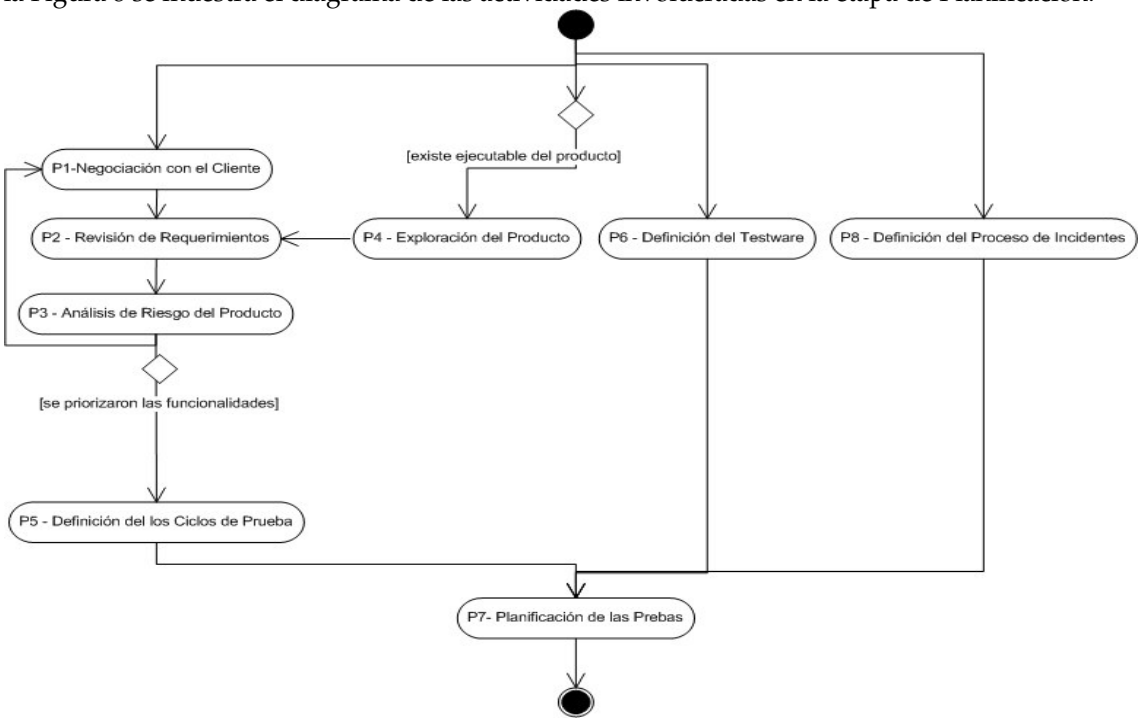

Figura 8 - Actividades en la Etapa: Planificación

. La actividad P1- Negociación con el Cliente tiene como objetivo definir todas las funcionalidades a probar y la prioridad con que será probada cada una, junto con el cliente. Para esto, en la actividad P2 – Revisión de Requerimientos se estudian las funcionalidades del producto y en P3 – Análisis de Riesgo del Producto, se realiza para conocer la importancia de incluir o no una funcionalidad en el alcance de las pruebas y la prioridad con que debe ser probadas en caso de ser incluida. A partir de esta información, se realiza P5 – Definición de los Ciclos de Prueba, donde se agendan los ciclos de prueba según las versiones que serán generadas por el equipo de desarrollo y se definen las funcionalidades que serán probadas en cada ciclo de prueba.
En el caso de que ya exista una versión ejecutable del producto al comenzar el proyecto de prueba, se puede realizar la actividad P4 – Exploración del Producto, donde se explora el producto con el fin de entender rápidamente los requerimientos y ayudar en la actividad P2 – Revisión de Requerimientos.
En paralelo con esto, se desarrollan dos actividades: P6 – Definición del Testware (el testware es el producto resultante de la prueba), donde se define cuales serán los elementos que lo conformarán para el proyecto y cómo se organizarán, P8 – Definición del Proceso de Incidentes, donde se define junto a el cliente y los desarrolladores la herramienta a seguir para el reporte de incidentes y las etapas involucradas en dicho proceso.
El plan de pruebas del proyecto es realizado en la actividad P7 – Planificación de las Pruebas. Dicho plan resume toda la información del proyecto de prueba y las decisiones tomadas durante la etapa de Planificación.

4.2.3 Ciclo de Prueba

El objetivo de esta etapa es realizar las pruebas para una versión determinada del producto. En un ciclo de prueba se puede ejecutar una, alguna o todas las pruebas planificadas para el producto. Esta etapa se divide a su vez en sub-etapas como se muestra en la Figura 9. Las sub-etapas son: Seguimiento del Ciclo, Configuración del Entorno, Diseño de las Pruebas y Ejecución de las Pruebas.

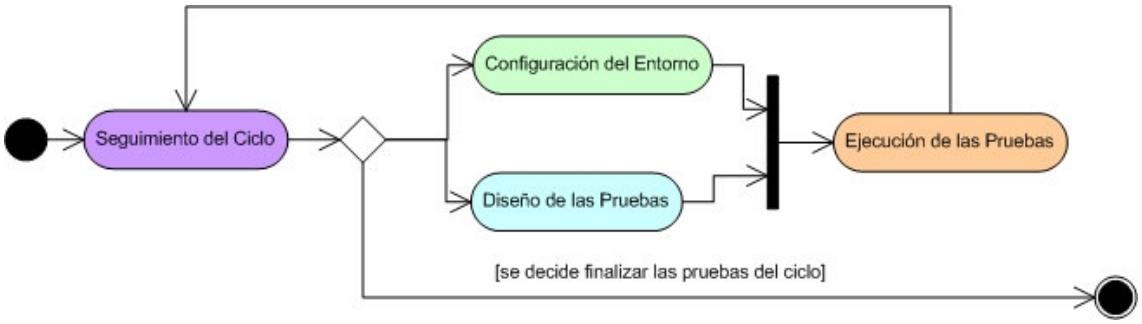

Figura 9 - Sub-etapas en cada Ciclo de Prueba

Los objetivos de cada sub etapa se describen a continuación:

Seguimiento del Ciclo: El objetivo es realizar el seguimiento y control del ciclo de prueba. La planificación realizada al principio del proyecto es revisada al comenzar cada ciclo de prueba, se planifican en detalle las tareas para el ciclo y se ajusta la planificación según las estimaciones y posteriores mediciones realizadas en los ciclos anteriores. Luego que se realizaron las otras tres sub-etapas del ciclo, se vuelve a esta sub-etapa, donde se evalúa el ciclo y se decide si se finaliza o si es necesario generar y ejecutar nuevas pruebas para el ciclo.

Configuración del Entorno: El objetivo de esta sub-etapa es configurar el ambiente de pruebas, separando los ambientes de desarrollo y prueba, instalar las herramientas necesarias y el software a probar en la versión correspondiente a cada ciclo de prueba.

Diseño de las Pruebas: Consiste en el diseño de los casos de prueba a partir de la especificación del producto. En caso de ser necesario, en esta sub-etapa se mejoran o generan las especificaciones junto con el Cliente.

Ejecución de las Pruebas: El objetivo de esta etapa es contrastar el comportamiento esperado del software con su comportamiento real, analizar las diferencias y reportar los resultados.

En cada ciclo, se comienza con la sub-etapa: Seguimiento del Ciclo, donde se administra y planifica el ciclo de prueba, una vez definidas las tareas que serán realizadas en el ciclo, se comienza la sub-etapa Diseño de las Pruebas, apenas se cuente con la versión ejecutable del producto a probar en este ciclo, comienza la sub-etapa de Configuración del Entorno, a continuación se realiza la Ejecución de las Pruebas.
Si no se cuenta con la versión ejecutable del producto, el Diseño de las Pruebas puede realizarse de todos modos, ya que se realiza a partir de la especificación del producto, en cambio la Configuración del Entorno y la Ejecución de las Pruebas requieren de la versión ejecutable del producto.
Al terminar de ejecutar las pruebas, se vuelve a la sub-etapa Seguimiento del Ciclo, donde se evalúan los resultados de la ejecución de las pruebas y se decide si es necesario derivar nuevos casos de prueba, en caso contrario se termina el ciclo de prueba.
Los ciclos de prueba se solapan en el tiempo, mientras se ejecutan las pruebas para un ciclo, se puede al mismo tiempo estar diseñando las pruebas del ciclo siguiente. En cada ciclo se revisan solamente las especificaciones de las funcionalidades que serán probadas en ese ciclo, con el objetivo de poder diseñar las pruebas que serán ejecutadas en ese ciclo.
A continuación se describen cada una de las sub-etapas para un Ciclo de Prueba.

Seguimiento del Ciclo

El objetivo de esta sub-etapa es realizar el seguimiento y control del proyecto de prueba en cada ciclo de prueba. En la Tabla 5 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Actividades	Artefactos generados
S1 – Planificación del Ciclo	PPC – Plan de Pruebas del Ciclo
P3 – Análisis de Riesgo del Producto	IP – Inventario de Pruebas
S2 – Administración de la Configuración	RC – Reporte de Configuración
S3 – Seguimiento y Control del Ciclo	RA – Reporte de Avance del Ciclo
S4 – Evaluación del Ciclo de Prueba	RE – Reporte de Evaluación del Ciclo
G1 – Estimación de las tareas	ET – Estimación de Tareas
G2 – Reportar el Esfuerzo	PE – Planilla de Esfuerzo

Tabla 5 – Seguimiento del Ciclo

En la Figura 10 se muestra el diagrama con las actividades de la sub-etapa Seguimiento del Ciclo.

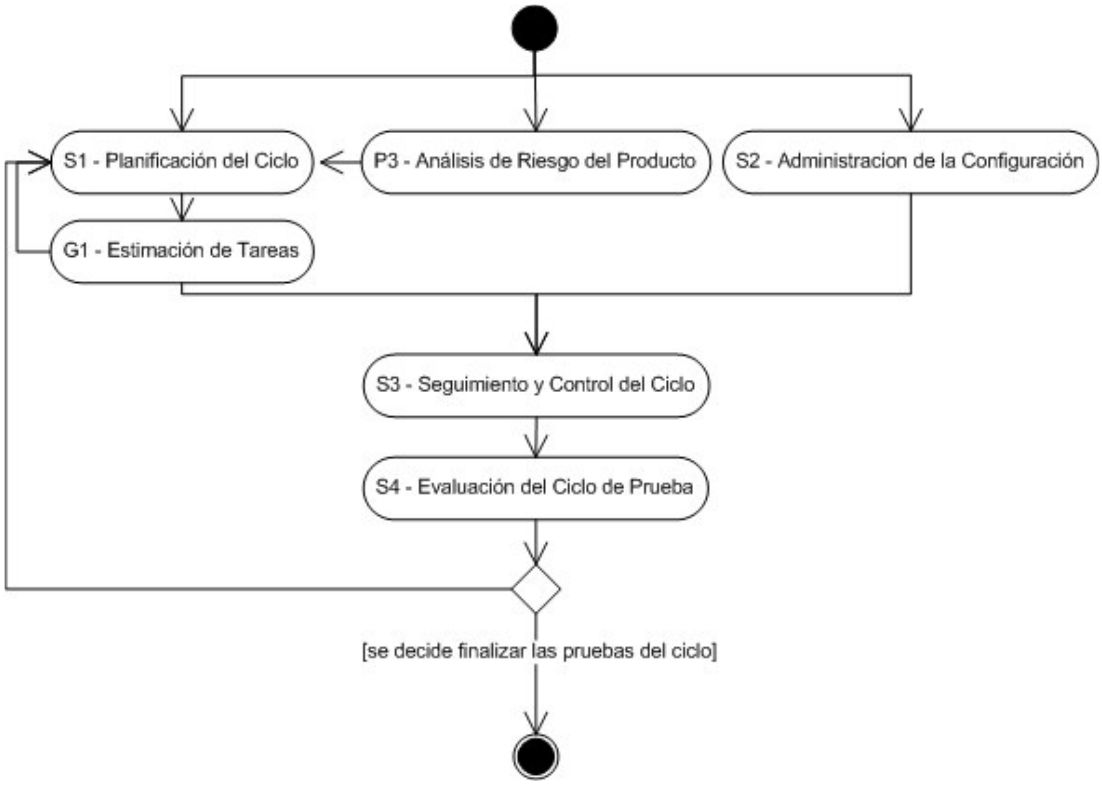

Figura 10- Actividades en la sub-etapa Seguimiento del Ciclo

La planificación del proyecto de prueba realizada en la etapa de Planificación, es revisada al comenzar cada ciclo en la actividad S1-Planificación del Ciclo. El detalle de la planificación se hace a partir de las estimaciones realizadas en la actividad G1-Estimación de Tareas. En la actividad P3- Análisis de Riesgo del Producto se definen las funcionalidades a probar en el ciclo. Durante todo el ciclo de prueba, se administra la configuración de los elementos que componen el testware en S3 – Administración de la Configuración y se realiza S5 – Seguimiento y Control del Ciclo. Al finalizar el ciclo de prueba se evalúa en la actividad S6-Evaluación de las Pruebas si es necesario derivar nuevos casos de prueba o si se termina el ciclo.

Configuración del Entorno

El objetivo de esta sub-etapa es configurar el ambiente de pruebas, separando los ambientes de desarrollo y prueba e instalar las herramientas necesarias y el software a probar en la versión correspondiente a cada ciclo de prueba. En la Tabla 6 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Actividades	Artefactos generados
C1 –Instalación de Herramientas	RO – Reporte de Obstáculos
C2 – Instalación y Configuración	RO – Reporte de Obstáculos

Tabla 6 – Configuración del Entorno

En la Figura 11 se muestra el diagrama de las actividades de la sub-etapa Configuración del entorno. En la actividad C1-Instalación de Herramientas, se instalan las herramientas necesarias para realizar las pruebas del producto, las cuales fueron definidas en la etapa de Planificación en la actividad P6-Definición del Testware. En el momento en que se cuenta con la versión ejecutable del producto a probar en el ciclo, se instala y se configuran los datos relacionados en la actividad C2- Instalación y

Configuración. Los problemas que puedan surgir durante estas actividades, se reportan en RO – Reporte de Obstáculos.

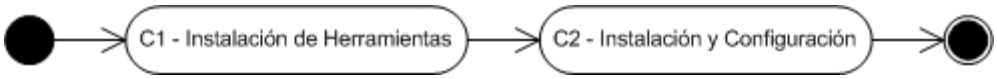

Figura 11 – Actividades de la sub-etapa Configuración del Entorno

Diseño de las pruebas

El objetivo de esta sub-etapa es diseñar las pruebas que serán luego ejecutadas sobre el producto a probar. En la Tabla 7 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Actividades	Artefactos generados
P2 – Revisión de Requerimientos	IP – Inventario de Pruebas
D1 – Diseño de los Casos de Prueba	CP – Casos de Prueba MT – Matriz de Trazabilidad
D2 – Validación de los Casos de Prueba	RR – Resumen de Reunión
D3 – Asignación de los Casos de Prueba	PE – Plan de Ejecución

Tabla 7 – Diseño de las Pruebas

En la Figura 12 se muestra el diagrama de las actividades a realizar en la sub-etapa de Diseño de las Pruebas. Se comienza con la actividad P2- Revisión de Requerimientos, donde se revisan solo las especificaciones de las funcionalidades que serán probadas en este ciclo, en caso de que no se cuente con las especificaciones o que estén incompletas, se trabaja junto con el cliente para generarlas o mejorarlas. A partir de los requerimientos para las pruebas se diseñan los casos de prueba en la actividad D1-Diseño de los Casos de Prueba, luego la actividad D2-Validación de los Casos de Prueba, tiene como objetivo que el cliente valide los casos de prueba generados para asegurarse de que son de valor para el negocio. Por último en la actividad D3-Asignación de los Casos de Prueba, quienes diseñan los casos de prueba, se los asignan a los testers encargados de ejecutarlos en la siguiente sub-etapa: Ejecución de las Pruebas.

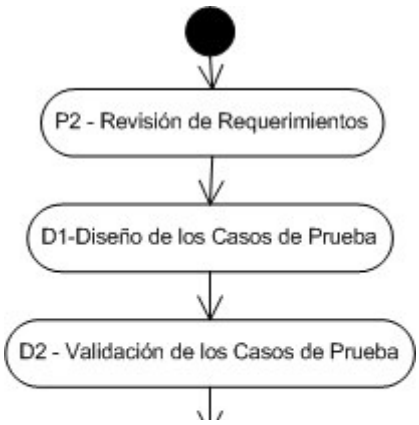

Figura 12 – Actividades de la sub-etapa Diseño de las Pruebas

Ejecución de las Pruebas

El objetivo de esta sub-etapa es contrastar el comportamiento esperado del software con su comportamiento real, analizar las diferencias y reportar los resultados. En la Tabla 8 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Actividades	Artefactos generados
E1 – Pruebas de Humo	RP – Reporte de Ejecución de las Pruebas
E2 – Ejecución de las Pruebas	RP – Reporte de Ejecución de las Pruebas
E3 – Testing Exploratorio	RP – Reporte de Ejecución de las Pruebas
E4 – Reporte de Incidentes	RI – Reporte de Incidente
E5 – Validación de los Incidentes	RI – Reporte de Incidente
E6 – Verificación de las Correcciones	RP – Reporte de Ejecución de las Pruebas

Tabla 8 – Ejecución de las Pruebas

En la Figura 13 se muestra el diagrama con las actividades de la sub-etapa Ejecución de las Pruebas, comienza con la actividad E1- Pruebas de Humo, donde se valida que las funcionalidades básicas de la versión a probar están presentes, si esto sucede se continúa con la actividad E2 – Ejecución de las Pruebas y E3- Testing Exploratorio, donde se ejecutan las pruebas del producto. Para los incidentes encontrados en ciclos anteriores, que fueron corregidos para la versión que se está probando, se ejecutan las pruebas de regresión en la actividad E6- Verificación de las Correcciones. Los incidentes encontrados en estas actividades se reportan en E4- Reporte de Incidentes. Los incidentes reportados son validados por el cliente y los desarrolladores en la actividad E5 – Validación de los Incidentes.

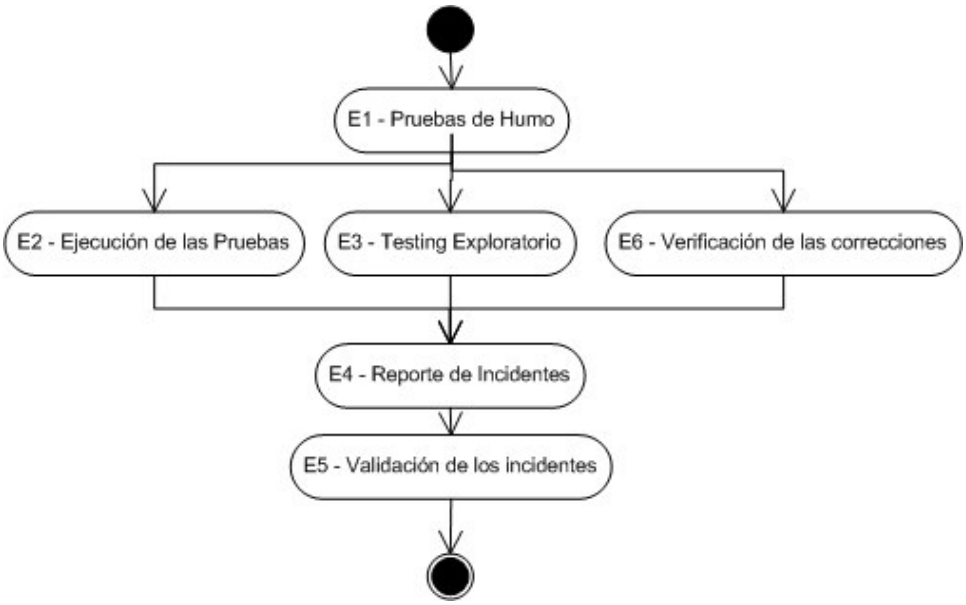

Figura 13– Actividades de la sub-etapa Ejecución de las Pruebas

4.2.4 Evaluación del Proyecto

Esta etapa tiene como objetivo conocer el grado de satisfacción del cliente, realizar el informe final, evaluar el proceso de prueba para su mejora y almacenar los elementos del proyecto de prueba para su

uso en proyectos posteriores. En la Tabla 9 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Actividades	Artefactos generados
V1- Evaluación de la Satisfacción del Cliente	ISC – Informe de Satisfacción del Cliente
V2 – Reporte Final del Proyecto	RF – Reporte Final del Proyecto de Prueba
V3 – Archivar el Testware	TW – Testware
V4 – Ajustes y Mejoras del Proceso de Prueba	APT – Ajustes al Proceso de Prueba
G1 – Estimación de las tareas	ET – Estimación de Tareas
G2 – Reportar el Esfuerzo	PE – Planilla de Esfuerzo

Tabla 9 – Evaluación del Proyecto

En la Figura 14 se muestra el diagrama de actividades para la etapa Evaluación del Proyecto. La actividad V1- Evaluación de la Satisfacción del Cliente tiene como objetivo conocer el grado de satisfacción del cliente con el proyecto de prueba, a partir de este dato en la actividad V2 – Mejoras y Ajustes al Proceso de Prueba, el equipo de prueba mejora el proceso para próximos proyectos, en la actividad V3- Reporte Final del Proyecto, se realiza el informe final del proyecto de prueba y en V4- Archivar el Testware se almacenan los elementos del Testware para ser consultados en próximos proyectos.

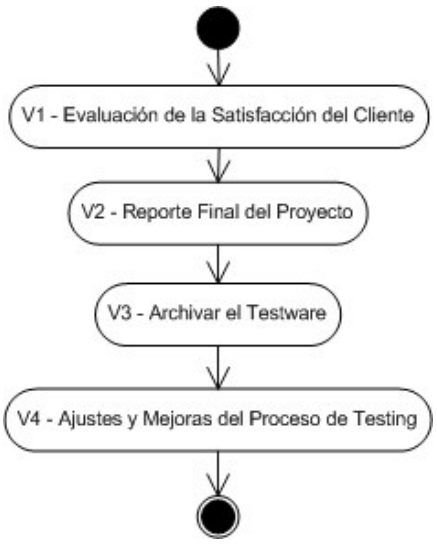

Figura 14– Actividades de la etapa Evaluación del Proyecto

4.3 Roles

Para ProTest se optó por los mismo roles y responsabilidades que los definidos para el Centro de ensayos de Software. Los roles definidos son: Líder de Proyecto de Prueba, Diseñador de Pruebas, Tester, Cliente y desarrollador o contraparte técnica.

El equipo de prueba está conformado por:

- El Líder de Prueba
- Los Diseñadores de Pruebas
- Los Testers.

El líder es quien dirige el proyecto de prueba. El diseñador es quien realiza el diseño de los casos de prueba a partir de las especificaciones del producto y el Tester es quien ejecuta los casos de prueba y reporta los resultados.
En cada proyecto de prueba, existe un cliente que es quien contrata el servicio de prueba. Un proyecto de prueba independiente puede ser contratado por:

- Quien va a comprar el producto de software
- Quien desarrolla el producto de software

En el primer caso, el Cliente es el mismo para el equipo de pruebas y para la empresa que desarrolla el producto. El Cliente puede contratar el equipo de pruebas independiente porque quiere asegurarse que el producto cumple con las especificaciones. En el segundo caso, la empresa que desarrolla el software decide contratar al equipo de pruebas independiente, esto puede ocurrir por ejemplo porque la empresa no tiene un equipo de pruebas propio. En dicho caso el rol del Cliente es desempeñado seguramente por un gerente o el líder de proyecto de desarrollo.

Los requisitos, competencias y responsabilidades para cada uno de los roles de ProTest se describen en el Capítulo 6 de este trabajo.

4.4 Relación con el ciclo de vida de Desarrollo

En esta sección se presenta la relación entre las etapas de ProTest y las etapas del ciclo de vida de desarrollo. Se asume un proceso iterativo e incremental para el desarrollo, ya que como se discutió en el Capítulo 3, es la forma más común de desarrollo actualmente. De todos modos, ProTest es independiente del ciclo de desarrollo, se asume en esta sección un proceso para desarrollo solo a modo ilustrativo.

Se discute cómo es la relación entre ProTest y el proceso de desarrollo cuando el proyecto de prueba comienza a la vez que el proyecto de desarrollo del producto o cuando el proyecto de prueba comienza estando el proyecto de desarrollo del producto en una etapa más avanzada o ya se encuentra instalado en el ambiente de producción y se está realizando mantenimiento del mismo.

La Figura 15 muestra el ciclo de vida de desarrollo en iteraciones, donde en cada iteración se realizan las etapas de: Requerimientos (R), Diseño (D), Implementación (I) y Prueba (P). La etapa de prueba en este caso se refiere a la prueba unitaria y de integración realizadas por el equipo de desarrollo. En cada iteración se genera una nueva versión del producto.

En la misma figura se muestran las etapas de ProTest acompañando el ciclo de desarrollo, en este caso la etapa Estudio Preliminar (EP) se hace antes de comenzar el desarrollo del producto, la etapa de Planificación (P) se realiza en paralelo con el análisis de requerimientos, ya que requiere realizar un primer análisis de riesgo del producto para definir los ciclos de prueba.
Una vez planificado el proyecto de prueba, los ciclos de prueba acompañan el desarrollo de la siguiente forma:

- La sub-etapa Diseño de las Pruebas (DP) se realiza a partir del análisis de requerimientos realizado por el equipo de desarrollo. El diseño de las pruebas se puede hacer en paralelo con las etapas de Diseño, Implementación y Prueba del ciclo de vida de Desarrollo. Cuando se libera la versión 1, se realiza la sub etapa Configuración del Entorno (CE) donde se instala la versión en el ambiente de prueba, luego de lo cual comienza la etapa Ejecución de las Pruebas (EJ). Durante todo el ciclo de prueba se realiza la sub-etapa Seguimiento del Ciclo.
- Para que los incidentes encontrados durante las pruebas sean de utilidad para el equipo de desarrollo, es conveniente que la ejecución de las pruebas de la primera versión termine antes

que la implementación de la versión 2. De esta forma, los desarrolladores pueden reparar los incidentes encontrados antes de liberar una nueva versión.

- Como se observa en la Figura 15, los ciclos de prueba se solapan en el tiempo, ya que mientras se están ejecutando las pruebas de una versión se puede realizar a la vez el diseño de las pruebas del ciclo siguiente.
- Al terminar el proyecto de prueba se realiza la etapa Evaluación del Proyecto (EV).

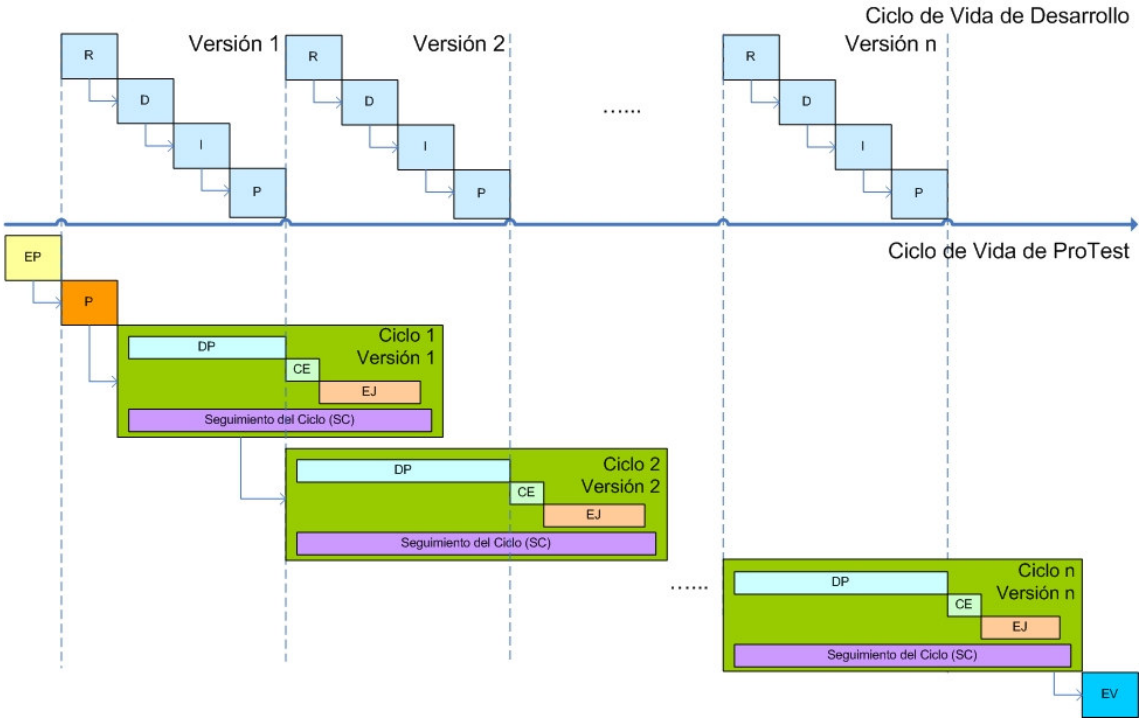

Figura 15– El proyecto de prueba acompaña el desarrollo

ProTest es lo suficientemente genérico como para ser aplicado en cada proyecto de prueba, por lo cual al comenzar cada proyecto, el proceso debe ser adaptado según las características particulares del mismo. A continuación se describen algunos hechos que suceden cuando el proyecto de prueba acompaña el proyecto de desarrollo. Es importante tenerlos en cuenta en el momento de realizar la planificación del proyecto de prueba:

- El equipo de prueba puede acceder a los documentos de requerimientos desde su concepción, pudiendo realizar sugerencias de mejoras a las especificaciones, de forma que se puedan derivar de ellas los casos de prueba.
- Debido a que en las primeras iteraciones el producto puede estar muy inestable, los incidentes reportados por el equipo de prueba, pueden haber sido ya encontrados y solucionados por desarrollo.
- Desarrollo genera muchas versiones en las primeras etapas, se debe definir cuales serán probadas y cuales no.
- Dado que los requerimientos en las primeras iteraciones de desarrollo se encuentran inestables, pueden sufrir muchos cambios, con la consecuencia de que muchos casos de prueba generados podrían tener que desecharse
- Los datos de prueba son difíciles de generar al principio, se debe trabajar con el cliente para definir datos de prueba lo más realista posibles.

- debido a que el producto esta constantemente cambiando, en los primeros ciclos de prueba se encuentran muchos incidentes, por lo que en la siguiente versión se deberán ejecutar muchas pruebas de regresión.
- El plan de desarrollo varía bastante a lo largo del proyecto, por lo que la planificación de los ciclos de prueba puede cambiar también.

El proyecto de prueba independiente puede ser contratado cuando el producto ya está bastante avanzado en su desarrollo o se encuentra en mantenimiento. En este caso, el proyecto de prueba comienza cuando se esta en una versión dada de desarrollo. En este caso se tiene que tener en cuenta que el equipo de prueba debe aprender sobre el producto.

A continuación se describen algunos hechos que suceden cuando el proyecto de prueba comienza en una etapa avanzada del desarrollo o en mantenimiento. Es importante tenerlos en cuenta en el momento de adaptar ProTest y realizar la planificación de cada ciclo:

- En caso de que no existan especificaciones o estén incompletas, estas pueden ser mejoradas, usando como fuente de requerimientos la última versión del producto que se encuentre disponible.
- En caso de que el producto se encuentre en mantenimiento, se cuenta seguramente también con manual de usuario, el cual es útil para entender el producto a probar y mejorar los requerimientos en caso de ser necesario.
- Si el producto esta en mantenimiento, se cuenta con datos de producción para generar los datos de prueba y se conocen los escenarios de uso mas populares.
- Es importante incluir en cada ciclo de prueba las funcionalidades que fueron cambiadas, ya que arreglos y modificaciones son riesgos frescos [KBP01]. Se requiere la ayuda del cliente y desarrolladores para realizar el análisis de impacto de un cambio y poder decidir que pruebas de regresión se deben ejecutar.
- Dado que las versiones del producto son más estables, en general, la planificación de desarrollo se cumple, entregando las versiones en la fecha acordada para comenzar el ciclo de prueba.
- En el caso de que se haya contratado la prueba independiente poco tiempo antes de instalar el producto en el ambiente del cliente, el equipo de prueba va a tener una gran presión por realizar las pruebas y encontrar los incidentes antes de generar una nueva versión del producto.
- También el equipo de prueba puede recibir presión para probar una versión intermedia que no fue planificada en los ciclos de prueba, pero que arregla un incidente encontrado durante las pruebas, en ese caso se debe negociar nuevamente con el cliente la agenda de ciclos y las prioridades en cada uno.

Capítulo 5

ACTIVIDADES DE PROTEST

Las Actividades en ProTest se organizan en etapas: Estudio Preliminar, Planificación, Ciclo de Prueba y Evaluación, como se describió en el Capítulo 4.

En la sección 5.1 se define el concepto de actividad y se describe la notación usada para las actividades en este capítulo. En la sección 5.2 se describen las actividades para la etapa Estudio Preliminar, en la sección 5.3 se definen las actividades para la etapa de Planificación, en la sección 5.4 se describen las actividades de la etapa Ciclo de Prueba, en la sección 5.5 se describen las actividades de la etapa Evaluación y en la sección 5.6 se describen las actividades comunes a todas las etapas como los son la estimación de tareas y el registro del esfuerzo.

5.1 Actividades

Una actividad es un conjunto de acciones que se realizan con el objetivo de crear o actualizar uno o varios artefactos. Cada actividad se compone de un conjunto de artefactos de entrada que son las precondiciones para su ejecución, un conjunto de roles que son los responsables de que la actividad se realice, una descripción de las tareas a realizar en la actividad y un conjunto de artefactos de salida que son las poscondiciones de su ejecución [DP06].

En cada etapa las actividades tienen un flujo de ejecución definido para su realización, como se mostró en el Capítulo 4. Algunas actividades pueden hacerse en paralelo, y según el momento del proyecto en que se esté, se podrán realizar todas, algunas o ninguna de las actividades definidas siguiendo el flujo establecido.

Las actividades se organizan según las etapas de ProTest definidas en el Capítulo 4. Existen algunas actividades que se realizan en más de una etapa, por lo que las tareas a realizar en la actividad pueden cambiar de una etapa a otra.

Las actividades de ProTest se componen de los siguientes elementos:

- **Objetivo:** Descripción del objetivo que se alcanza al realizar la actividad
- **Descripción:** Descripción de cómo la actividad debe ser realizada
- **Figura:** Cada actividad viene acompañada de una figura, como la que se muestra en la Figura 16– I1 Definición del Servicio. La figura describe en forma gráfica las relaciones que tiene la actividad con los roles involucrados en que la actividad se realice y los artefactos que consume y genera la actividad. En la Tabla 10 se muestran los elementos presentes en la figura. Estas figuras son realizadas con la herramienta DeVeloPro, la cual se describe en el Anexo A.

Elemento	Icono
Actividad	
Roles	
Artefactos	

Tabla 10 – Elementos de la Actividad

Además existen las siguientes relaciones entre elementos:

- **Rol responsable:** Cada actividad tiene un rol responsable de que se realice, en la figura esto se muestra con una flecha roja desde el rol hacia la actividad.
- **Rol involucrado:** Son los roles que ayudan a realizar la actividad, en la figura se muestra con una flecha negra desde el rol a la actividad
- **Artefactos de entrada:** Son aquellos cuya flecha va desde el artefacto a la actividad
- **Artefactos de salida:** Son aquellos cuya flecha va desde la actividad al artefacto
- **Artefactos de entrada y salida:** Son aquellos cuya flecha es bidireccional entre el artefacto y la actividad.

A continuación se describen las actividades por etapa: Estudio Preliminar, Planificación, Ciclo de Prueba y Evaluación

5.2 Estudio Preliminar

Esta etapa esta descrita en el Capítulo 4. A continuación se describen las actividades definidas para la etapa Estudio Preliminar.

I1 – Definición del Servicio

Objetivo

Definir el servicio que brindará el equipo de pruebas independiente. Incluye definir el alcance de las pruebas del producto y una agenda global del proyecto de prueba. En la Figura 16 se muestran los artefactos y roles involucrados en esta actividad.

Figura 16– I1 Definición del Servicio

Descripción

Esta actividad se realiza varias veces durante la etapa Estudio Preliminar para definir el alcance del proyecto de prueba, esto es, qué funcionalidades se van a probar y cuales no. Se lleva a cabo en reuniones con el Líder de Proyecto, Cliente y Desarrolladores. Para poder definir el alcance, se divide el sistema en componentes o subsistemas, no todos los componentes serán probados con la misma importancia y pueden existir componentes que queden fuera del alcance de las pruebas. Cada componente agrupa varias funcionalidades, se dividen las funcionalidades hasta un nivel en el que sea posible definir el alcance. Luego de esto, en las actividades I2 – Revisión Preliminar de Requerimientos se analizan las funcionalidades, dando como resultado el IP-Inventario de Pruebas y en I3 – Análisis Preliminar de Riesgo se priorizan. Cuando se tiene el inventario priorizado de los grupos de funcionalidades, se retoma la actividad I1 – Definición del Servicio. Para definir el alcance es necesario conocer también el PD-Plan de Desarrollo del producto, donde se encuentra la planificación de las versiones del producto.
Como resultado de esta actividad se obtiene PS-Propuesta de Servicio que incluye el alcance, la cantidad de ciclos de prueba y la agenda preliminar del proyecto de prueba.
Otros puntos importantes que se deben definir en esta actividad para poder realizar la propuesta de servicio, incluyen:

- Definir dónde se realizará la ejecución de las pruebas, si en el laboratorio del equipo de prueba, en las instalaciones del cliente o se usará una modalidad de trabajo mixta.
- Definir los atributos de calidad externa que serán tenidos en cuenta para la prueba del producto, según el modelo de calidad ISO/IEC 9126 [ISO9126]. Se debe definir cuales de estas características de calidad se debe verificar que estén presentes en el producto y con que relevancia deben ser verificadas.

- Definir el criterio de terminación de las pruebas, en ProTest esto se vincula con el cubrimiento de las funcionalidades, teniendo en cuenta el riesgo asociado a dichas funcionalidades, la cantidad de ciclos definidos y la agenda pautada.
- Negociar con el cliente el porcentaje de pruebas de regresión sobre el tiempo total de cada ciclo que se realizarán. Esto se requiere para poder planificar el proyecto de prueba.

En general se necesitan 2 o 3 reuniones para realizar esta actividad. Dependiendo del tamaño y complejidad del producto a probar, pueden requerirse reuniones adicionales. Se resumen los puntos tratados en cada reunión en RR-Resumen de la Reunión, el cual es enviado al cliente para su validación.

I2 – Revisión Preliminar de Requerimientos

Objetivo

Estudiar las especificaciones del producto para determinar si es posible generar los casos de prueba a partir de ellas. En caso de que no existan especificaciones o que estén incompletas, se debe definir junto con el Cliente la forma en que serán generadas o mejoradas. En la

Figura 17 se muestran los artefactos y roles involucrados en esta actividad.

Figura 17– I2 Revisión Preliminar de Requerimientos

Descripción

Al realizar pruebas funcionales se debe decidir si la salida observada al ejecutar el programa es la salida esperada. La salida esperada esta expresada en las especificaciones del producto, se deben validar los requerimientos para poder usarlos como oráculo para las pruebas.

Se revisa la correspondencia entre las distintas fuentes de requerimientos y se evalúa si es posible realizar el diseño de las pruebas a partir de ellas. Si esto no es posible, se debe definir la forma de trabajo para mejorar los requerimientos existentes en las siguientes etapas, junto al Cliente y los desarrolladores. Como resultado de esta actividad se obtiene el artefacto IP - Inventario de pruebas.

I3 - Análisis Preliminar de Riesgo del Producto

Objetivo

Obtener una lista priorizada de las funcionalidades más importantes y riesgosas, así como identificar las funcionalidades que no serán verificadas y el riesgo asociado. En la Figura 18 se muestran los artefactos y roles involucrados en esta actividad.

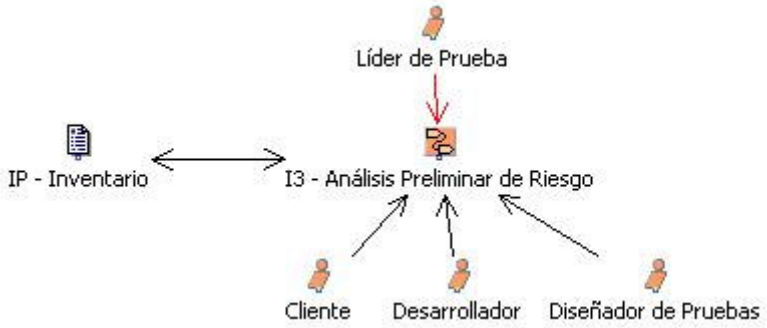

Figura 18– I3 Análisis Preliminar de Riesgo

Descripción

A partir del artefacto IP- Inventario de Prueba, se analizan los riesgos asociados a cada funcionalidad del inventario junto al cliente y desarrolladores. El cliente conoce las funcionalidades que presentan mayor riesgo para el negocio en caso de no funcionar correctamente y los desarrolladores conocen las partes del producto que es más probable que tengan defectos. Las funcionalidades del producto son priorizadas en función del análisis de riesgo realizado.

Este análisis se realiza sobre grupos de funcionalidades, el objetivo es obtener un mapa general de las funciones más importantes del producto y poder definir el alcance, realizar un primer cronograma del proyecto de prueba y obtener una cotización.

5.3 Planificación

Esta etapa está descrita en el Capítulo 4. A continuación se describen las actividades de la etapa de Planificación.

P1 - Negociación con el cliente

Objetivo

Negociar con el cliente el alcance definitivo de las pruebas y la agenda de los ciclos de prueba. En la Figura 19 se muestran los artefactos y roles involucrados en esta actividad.

Figura 19– P1 Negociación con el Cliente

Descripción

A partir de las funcionalidades de los componentes identificados en el IP-Inventario de Prueba durante el Estudio Preliminar, se refina cada componente, definiendo las funcionalidades en detalle junto al cliente.

Se revisa junto al cliente los atributos de calidad que serán tenidos en cuenta durante las pruebas, a partir de lo definido durante el Estudio Preliminar, donde se definieron los atributos y la relevancia de los mismos.

Se definen los productos resultantes de las actividades de prueba que serán entregados al cliente y en qué momento se entregará cada uno, también se define la forma de comunicación entre el cliente y el equipo de prueba, estas podrían ser reuniones semanales, presentación de informes semanales, reportes de incidentes diarios, entre otros.

Se identifican los responsables de cada parte implicada en el proyecto, al menos se debe conocer quienes son los responsables por parte del cliente de tomar las decisiones de alto nivel, quiénes contestarán dudas respecto al producto, quiénes manejarán los cambios del alcance de las pruebas a lo largo del proyecto, quiénes validarán los casos de prueba y los resultados de su ejecución.

Se deben negociar las responsabilidades en la gestión del ambiente de prueba, quién lo gestiona, quién instala nuevas versiones en el ambiente de prueba y quién provee los datos de prueba.

P2 - Revisión de Requerimientos

Objetivo

Estudiar las especificaciones del producto para decidir si es posible generar las pruebas a partir de ellas. En la Figura 20 se muestran los artefactos y roles involucrados en esta actividad.

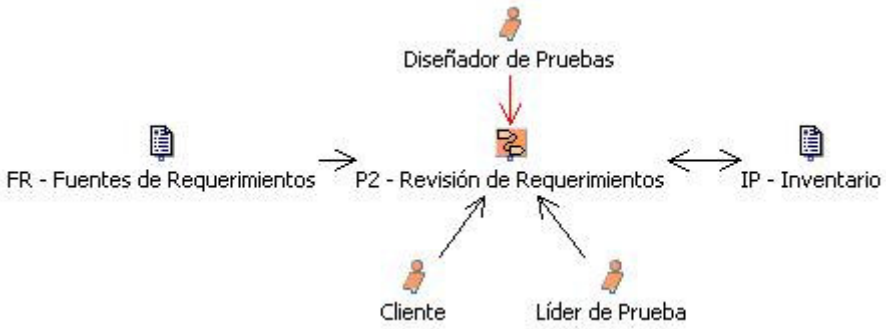

Figura 20 – P2 Revisión de Requerimientos

Descripción

A partir de la actividad P1- Negociación con el Cliente, donde se detallan las funcionalidades de cada componente, se analizan las especificaciones. Las especificaciones pueden ser obtenidas de diferentes fuentes: documentos de requerimientos, sistemas ya existentes, manuales del sistema, reportes de defectos, entrevistas con cliente y usuarios, prototipos.

Se revisan las especificaciones de las funcionalidades que van a ser probadas en el primer ciclo de prueba (esas funcionalidades serán probablemente las de mayor prioridad dada en el análisis de riesgo del producto, ya que revisar todos los requerimientos del sistema puede llevar demasiado tiempo). Se agenda la revisión de requerimientos de lo que resta en los sucesivos ciclos de prueba.

En caso de que sea necesario se mejoran los requerimientos y en el caso de que no se cuente con requerimientos, éstos deben ser especificados para poder realizar las pruebas a partir de ellos.

Si existe alguna versión ejecutable del sistema, la revisión de requerimientos se complementa con la actividad P4 -Exploración del Producto.

P3 - Análisis de Riesgo del Producto

Objetivo

Obtener una lista detallada de todas las funcionalidades, priorizada según importancia y riesgo. En la Figura 21 se muestran los artefactos y roles involucrados en esta actividad.

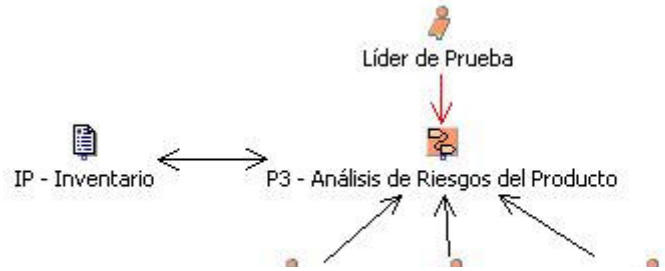

Figura 21– P3 Análisis de Riesgo del Producto

Descripción

Se toman los grupos de funcionalidades definidas en la etapa anterior y se realiza el análisis de riesgo de cada funcionalidad del producto, descomponiendo las funcionalidades a nivel macro definida en la etapa del Estudio Preliminar. Puede ocurrir que un grupo de funcionalidades al que se le asignó prioridad alta durante la etapa Estudio Preliminar, al ser dividido en varias funcionalidades, algunas de ellas sean de prioridad alta, otras de prioridad media y otras queden fuera del alcance de las pruebas.

P4 - Exploración del Producto

Objetivo

Explorar el producto para conocer rápidamente las funcionalidades que brinda. En la Figura 22 se muestran los artefactos y roles involucrados en esta actividad.

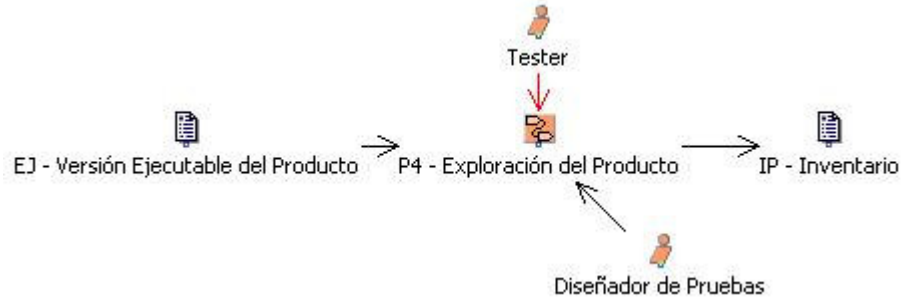

Figura 22– P4 Exploración del Producto

Descripción

Explorar el producto significa navegar con una misión general pero sin una ruta prefijada y involucra aprendizaje continuo y experimentación. Hasta que no se explora el producto, las pruebas son superficiales, incluso si se ha leído la especificación en detalle [KBP01].

Cuando se está realizando la evaluación de productos desarrollados por terceros, esta actividad ayuda a conocer el producto a probar. Esta actividad puede realizarse recién cuando se tiene una versión ejecutable del producto. El tester se familiariza con el producto mientras que el diseñador, puede comparar las especificaciones con el producto. Es un complemento a la actividad de P2 - Revisión de Requerimientos.

P5 - Definición de los ciclos de Prueba

Objetivo

Definir la agenda de los ciclos de prueba y las funcionalidades que serán probadas en cada ciclo, para todo el proyecto. En la Figura 23 se muestran los artefactos y roles involucrados en esta actividad.

Figura 23– P5 Definición de los Ciclos de Prueba

Descripción

A partir del nuevo IP –Inventario de Pruebas priorizado resultante de las actividades P2-Revisión de Requerimientos y P3-Análisis de Riesgos del Producto, se actualiza la agenda de ciclos de prueba realizada en la etapa anterior, detallando comienzo y fin de cada ciclo de prueba y su relación con las versiones del producto a probar. Se negociará con los desarrolladores el tiempo que necesitan para realizar las correcciones y generar la nueva versión a probar.

La decisión de qué pruebas ejecutar en cada ciclo se realiza a partir del análisis de riesgo del producto. Dado que en general no se tiene el tiempo como para volver a ejecutarlas todas para cada versión, se debe seleccionar un conjunto de pruebas en cada ciclo. Se pueden utilizar para esto las técnicas definidas en [Bla02]:

- Técnica de Priorización: Se ejecutan las pruebas en el orden de prioridad dado en el análisis de riesgo al comenzar las pruebas.
- Técnica de Priorización Dinámica: A medida que los ciclos se desarrollan pueden cambiar las prioridades, se ejecutan las pruebas en cada ciclo según el orden de prioridad definido en el ciclo. El riesgo con este enfoque es que podría ocurrir que no se ejecutan todas las pruebas a lo largo de los ciclos, lo que podría llevar a un cambio de agenda al final.
- Tren: Si todas las pruebas tienen la misma importancia se pueden ejecutar las pruebas una tras otra a través de los ciclos.

La peor situación es aquella donde cierto conjunto de pruebas que formaban parte del alcance nunca se ejecuta, o en la que éstas se ejecutan solamente una vez al principio y nunca se vuelven a ejecutar ya que quedan relegadas al venir las nuevas versiones. Esto podría pasar si se usa el enfoque del Tren comenzando en cada ciclo con la primera prueba otra vez o con la Técnica de Priorización Dinámica si ciertas funcionalidades nunca son tenidas en cuenta como importantes.

Al obtener una nueva versión donde se corrigieron defectos, se deben ejecutar nuevamente las pruebas de regresión para esos defectos. Como a priori no se conoce cuales ni cuantos serán esos defectos, al comenzar cada ciclo, deben ser planificadas las pruebas de regresión.

P6 - Definición del Testware

Objetivo

Definir los artefactos que serán utilizados en el proyecto de prueba y cómo se organizarán. En la Figura 24 se muestran los artefactos y roles involucrados en esta actividad.

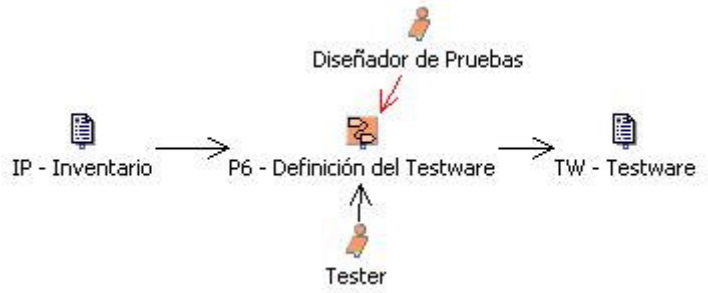

Figura 24– P6 Definición del Testware

Descripción

Definir la estructura del testware y cómo se organizarán y accederá a los elementos del mismo. El testware debe ser accedido frecuentemente, por lo que los casos de pruebas deben ser organizados y catalogados para su acceso. Puede ser útil agrupar los casos de pruebas que requieren el mismo entorno de ejecución o las pruebas que requieren mantenimiento y deben ser examinadas previo a su uso. Para cada caso de prueba se debe indicar datos de pruebas, procedimiento de prueba y archivos asociados (fuente, objeto, datos, forma de ejecutar, reporte) [Kit95].

P7 - Planificación de las pruebas

Objetivo

Elaborar el plan de pruebas del proyecto, donde se define quién, cuándo, dónde y cómo se realizan las actividades de prueba. En la Figura 25 se muestran los artefactos y roles involucrados en esta actividad.

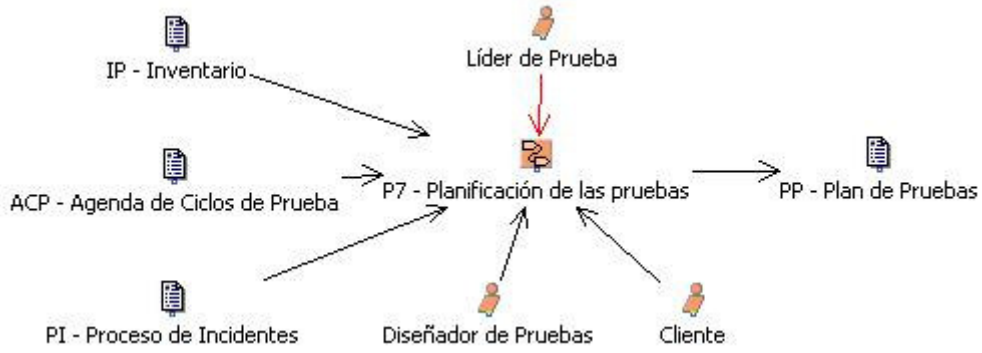

Figura 25– P7 Planificación de las Pruebas

Descripción

Realizar el plan de pruebas brinda la posibilidad de recolectar las ideas y cristalizarlas en tareas concretas, puede ser visto como un medio de comunicación entre el equipo de pruebas y el cliente [Bla02].
En el plan de pruebas se incluye:

- alcance del proyecto
- calendario
- ciclos y las condiciones que deben cumplirse para comenzar cada uno
- requerimientos necesarios de hardware y software para ejecutar las pruebas

- estrategia de pruebas usada durante el proyecto
- técnicas usadas para el diseño de las pruebas
- roles y sus responsabilidades
- procedimientos a seguir para el reporte y seguimiento de incidentes
- análisis de los riesgos del proyecto de prueba
- clasificación de los defectos usada en el proyecto
- especificación de qué reportes y otros productos se entregarán al cliente durante el proyecto y al finalizar el mismo.

Para la planificación de la agenda se utiliza la lista de funcionalidades a probar en cada ciclo. En función de esto se estiman la duración de las actividades del proyecto de prueba. Para la estimación de las actividades se utilizará la información de proyectos anteriores referentes a actividades similares.

El plan de pruebas es realizado por el Líder de Proyecto y Diseñador, es validado por el Cliente.

P8 - Definición del Proceso de Incidentes

Objetivo

Definir los estados por los que pasa un incidente, desde que es encontrado hasta que es reparado en el producto. Incluye definir las responsabilidades. En la Figura 26 se muestran los artefactos y roles involucrados en esta actividad.

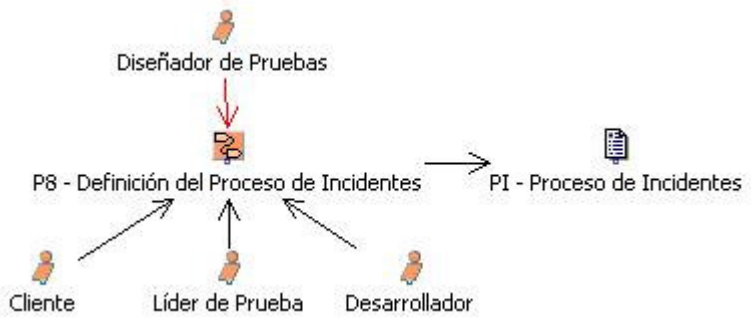

Figura 26– P8 Definición del Proceso de Incidentes

Descripción

Un incidente puede ser un defecto del producto, una mejora a realizar o una observación. El sistema de seguimiento de incidentes permite al equipo de prueba reportar, administrar y analizar los incidentes reportados y la tendencia de los mismos en cada ciclo de prueba. Esta actividad incluye:

- Definir la herramienta de seguimiento de incidentes a usar en el proyecto de prueba (puede ocurrir que el cliente ya tenga una herramienta de seguimiento y en ese caso, se podría utilizar la misma herramienta).
- Definir quienes de las personas de desarrollo van a ser los encargados de recibir los incidentes detectados y una vez corregidos, de qué forma serán retornados para su prueba de regresión.
- Definir los estados por los que pasará el incidente, en general como mínimo se tienen los siguientes estados [KFN99]:
  - El incidente es ingresado por un tester en el sistema de seguimiento de incidentes
  - El gerente de desarrollo le asigna una prioridad al incidente y se lo asigna a un desarrollador para ser reparado, o lo rechaza por no poder reproducirlo o entender que no es un incidente

- El incidente es reparado por un desarrollador, éste queda en el estado Reparado
- El tester vuelve a ejecutar las pruebas que encontraron el incidente, en caso de que haya sido solucionado, el incidente pasa al estado Resuelto, en caso contrario pasa al estado Pendiente

5.4 Ciclo de Prueba

El objetivo de esta etapa es realizar las pruebas para cada versión del producto. En un ciclo de prueba se puede ejecutar una, alguna o todas las pruebas planificadas para el producto. Esta etapa se divide a su vez en sub-etapas como se muestra en la Figura 9 de la sección 4.2.3 de este trabajo. Las sub-etapas son: Seguimiento del Ciclo, Configuración del Entorno, Diseño de las Pruebas y Ejecución de las Pruebas.

5.4.1 Seguimiento del Ciclo

Esta sub-etapa se describe en el Capítulo 4. A continuación se describen las actividades de la sub-etapa Seguimiento del Ciclo.

P3 - Análisis de riesgo del producto

Esta actividad fue descrita en la etapa de Planificación. En la Figura 21 se muestran los artefactos y roles involucrados en esta actividad.

Durante el Seguimiento del Ciclo, el IP-Inventario de Prueba resultante del análisis de riesgo del producto realizado durante la etapa de Planificación puede requerir ser ajustado, debido a cambios en las prioridades del negocio o a la confianza adquirida en el producto como resultado de la realización de las pruebas en ciclos anteriores. En cada ciclo, se revisa junto con el cliente y los desarrolladores la lista de riesgos y se modifica en caso de ser necesario.

S1 - Planificación del ciclo

Objetivo

Elaborar el plan de pruebas del ciclo, donde se define el detalle de las actividades a realizar y los involucrados en las mismas. En la Figura 27 se muestran los artefactos y roles involucrados en esta actividad.

Figura 27- S1 Planificación del Ciclo

Descripción

En esta actividad se realiza el plan detallado de las tareas a realizar en el ciclo de prueba, para esto se utiliza la estimación de tareas realizada en la actividad G1 – Estimación de Tareas que fueron plasmadas en ET-Estimación de Tareas, las mediciones de ciclos anteriores que se encuentran en RA-Reporte de Avance del Ciclo, el análisis de riesgo del producto realizado y las funcionalidades para el ciclo que se encuentran en IP-Inventario de Pruebas y el PP-Plan de Pruebas que contiene la planificación de todo el proyecto. Se debe tener en cuenta las pruebas de regresión que deben ejecutarse en el ciclo, debido a incidentes encontrados en ciclos anteriores que fueron corregidos para la versión de este ciclo.

S2 - Administración de la Configuración

Objetivo

Gestionar la configuración de los elementos que componen el testware. En la Figura 28 se muestran los artefactos y roles involucrados en esta actividad.

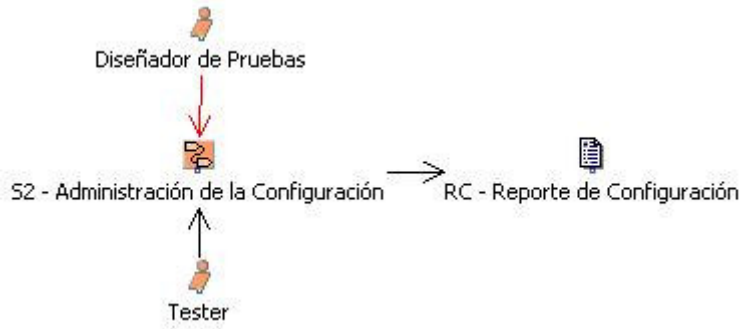

Figura 28- S2 Administración de la Configuración

Descripción

El objetivo de la gestión de la configuración es mantener la integridad de los productos que se obtienen a lo largo del desarrollo de un sistema, controlando los cambios que se realizan. El interés de las pruebas en la gestión de la configuración tiene que ver con administrar los elementos que componen el testware, asociar la versión de prueba con su correspondiente versión de software a probar, identificar bajo qué configuraciones de software y hardware se reportó un incidente de forma de poder reproducirlo y asegurarse de que se está realizando la prueba de la versión correcta [Kit95]. Esta actividad se realiza durante todo el ciclo de prueba, al finalizar el ciclo se realiza un reporte con el estado de los elementos que componen el testware para el ciclo.

S3 - Seguimiento y Control del Ciclo

Objetivo

Controlar que las actividades del ciclo se estén realizando según lo planificado. En la Figura 29 se muestran los artefactos y roles involucrados en esta actividad.

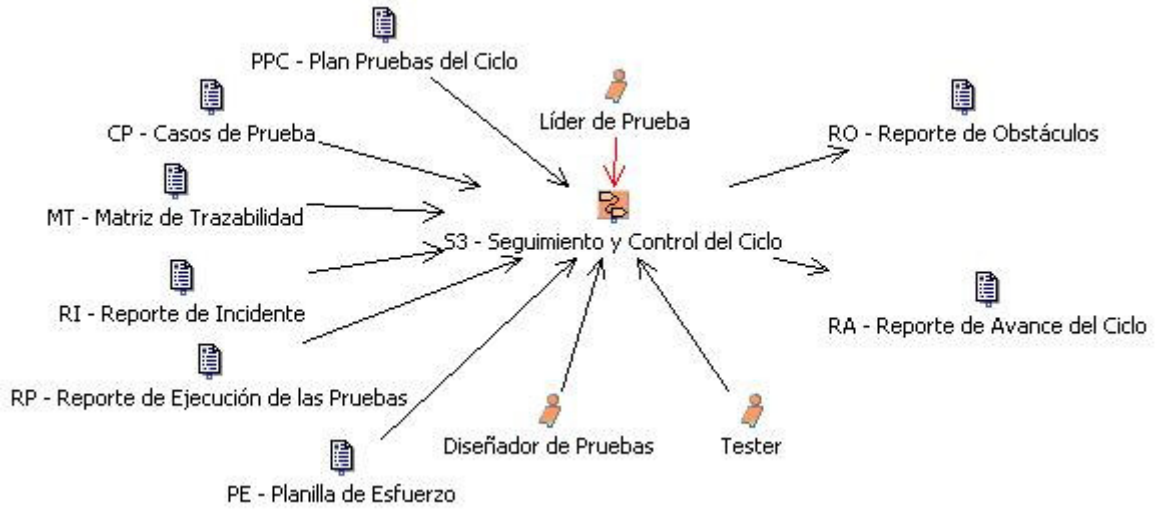

Figura 29– S3 Seguimiento y Control del Ciclo

Descripción

Esta tarea se realiza durante todo el ciclo de prueba, se controla que las tareas sean realizadas, y que los tiempos que insumen las tareas estén dentro de lo planificado. Se realiza contingencia y mitigación de los riesgos del proyecto de prueba.
Para realizar el seguimiento de las pruebas ejecutadas se puede contar el número de casos de prueba en cada una de las siguientes categorías [Kit95]

- Planificados: Casos de Prueba que se planifican realizar
- Disponibles: Casos de Prueba planificados que están disponibles para ser ejecutados
- Ejecutados. Casos de Prueba Disponibles que han sido ejecutados
- Exitosos: Casos de Prueba Ejecutados que en las ejecuciones más recientes no han detectado incidentes

Graficar estos 4 valores en el tiempo periódicamente ayuda a analizar las tendencias de las pruebas.

El control del ciclo implica tomar cualquier acción correctiva como resultado de la información y las mediciones obtenidas durante el seguimiento. Ejemplos de acciones a tomar para el control de las pruebas son [ISQ05]:

- Volver a priorizar las pruebas cuando un riesgo del proyecto ocurre
- Cambiar la agenda de las pruebas
- Agregar nuevos criterios de entrada antes de aceptar una nueva versión

A lo largo del proyecto pueden ocurrir obstáculos, como por ejemplo: problemas con la instalación de la versión. En RO-Reporte de Obstáculos se mantiene una lista de los problemas que ocurrieron en el ciclo.
Al finalizar el ciclo de prueba se realiza el artefacto RA-Reporte de Avance del Ciclo, mostrando el avance de las actividades de prueba realizadas y de la calidad de la versión probada. Se reporta cuánto fue probado el producto en varias dimensiones: incidentes, requerimientos, configuraciones, entre otras.

S4 - Evaluación de las Pruebas

Objetivo

Evaluar las pruebas del ciclo y decidir si se puede terminar el ciclo o si es necesario generar nuevas pruebas. En la Figura 30 se muestran los artefactos y roles involucrados en esta actividad.

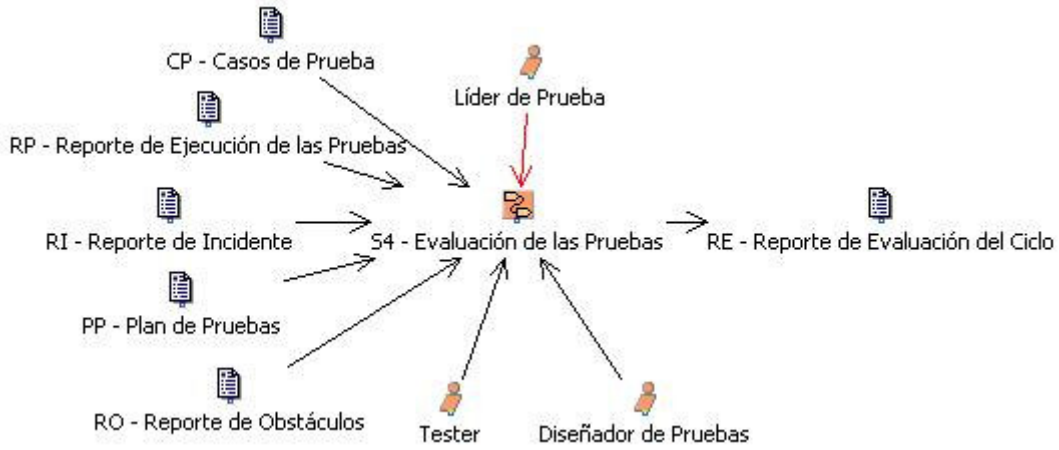

Figura 30- S4 Evaluación de las Pruebas

Descripción

Esta actividad ocurre luego de que se realizaron las otras tres sub-etapas del ciclo: Diseño de las Pruebas, Configuración del Entorno y Ejecución de las Pruebas.

La evaluación de las pruebas cubre los siguientes tres puntos [Kit95]:

- Evaluación del cubrimiento: Evaluar los casos de prueba según el cubrimiento de funcionalidades.
- Evaluación de los incidentes del producto: Evaluar la calidad del producto respecto a la ejecución de las pruebas realizadas.
- Evaluación de la efectividad de las pruebas: Evaluar las pruebas respecto al criterio de completitud y decidir cuando parar o agregar más pruebas y seguir, se basa en los dos anteriores.

Se deben responder las siguientes preguntas: ¿Continuar o parar las pruebas? ¿Qué pruebas adicionales son necesarias si se decide continuar?

5.4.2 Configuración del Entorno

Esta sub-actividad fue descrita en el Capítulo 4. A continuación se describen las actividades de la sub-etapa Configuración del Entorno.

C1 - Instalación de Herramientas

Objetivo

Instalar las herramientas que se usarán para la realización de las pruebas, las cuales fueron definidas en el Testware. En la Figura 31 se muestran los artefactos y roles involucrados en esta actividad.

Figura 31– C1 Instalación de Herramientas

Descripción

Se instalan las herramientas definidas en el testware para realizar las pruebas, en cada computadora personal y en los servidores. Dentro de las herramientas necesarias para realizar las pruebas se encuentran: sistemas operativos, herramientas para la automatización de las pruebas, manejadores de bases de batos, herramientas para la gestión de las pruebas, herramientas para la realización de reportes, sistema de gestión de incidentes, herramienta para la gestión de la configuración. En caso de encontrar problemas con la instalación, estos se reportan en RO-Reporte de Obstáculos.

C2 - Instalación y Configuración

Objetivo

Instalar la versión del producto a probar en el ciclo. En la Figura 32 se muestran los artefactos y roles involucrados en esta actividad.

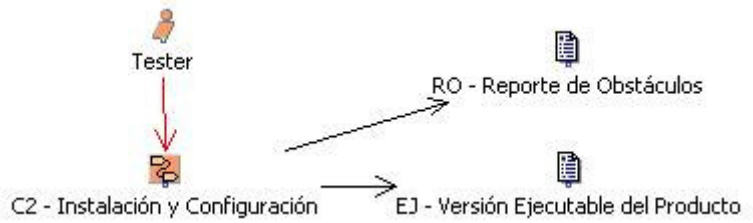

Figura 32– C2 Instalación y Configuración

Descripción

En cada ciclo se debe instalar en el ambiente de prueba la versión correspondiente del producto al que se le va a realizar la prueba funcional. Esto incluye configurar los datos generales relacionados. En caso de encontrar problemas con la instalación, estos se reportan en RO-Reporte de Obstáculos.

5.4.3 Diseño de las pruebas

Esta sub-actividad fue descrita en el Capítulo 4. A continuación se describen las actividades de la sub-etapa Diseño de las Pruebas.

P2 - Revisión de Requerimientos

Esta actividad fue descrita en la etapa de Planificación. En la Figura 20 se muestran los artefactos y roles involucrados en esta actividad.

En cada ciclo de prueba se revisan las especificaciones existentes de las funcionalidades que han sido planificadas probar para este ciclo, en caso de que no sea posible realizar el diseño de las pruebas a partir de ellas, se trabaja junto con desarrolladores y cliente en mejorar las especificaciones existentes.

D1 - Diseño de los Casos de Prueba

Objetivo

Derivar los casos de prueba a partir de las especificaciones del producto. En la Figura 33 se muestran los artefactos y roles involucrados en esta actividad.

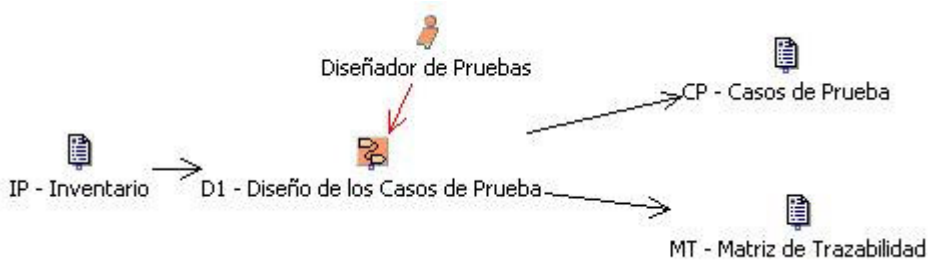

Figura 33– D1 Diseño de los Casos de Prueba

Descripción

A partir de las funcionalidades planificadas para el ciclo se desarrollan los casos de prueba usando técnicas de caja negra. Esta actividad incluye diseñar las pruebas e identificar los datos de prueba.

Para cada funcionalidad a probar, se crea una matriz que muestra la correspondencia entre el ítem de prueba y el caso de prueba, de forma de conocer qué casos de prueba cubren qué ítems de prueba [Kit95]. Entre las técnicas posibles de caja negra a utilizar para el diseño de los casos de prueba se encuentran: partición de equivalencia, valor límite, conjetura de errores, grafo causa efecto, partición en categorías, máquinas de estado finitas y escenarios, las cuales fueron resumidas en el capítulo 2.

En esta actividad se especifican los casos de prueba asociados a cada elemento que compone el IP-Inventario de Pruebas para este ciclo. Los pasos para realizarla incluyen [Kit95]:

- identificar los elementos que deben ser probados
- analizar las prioridades de esos elementos dadas por el análisis de los riesgos del producto
- desarrollar el diseño de las pruebas de alto nivel para grupos de pruebas relacionados
- desarrollar los casos de pruebas individuales a partir del diseño de alto nivel
- crear una matriz de cubrimiento que muestre la correspondencia entre las funcionalidades a probar y los casos de prueba

Se deben identificar los ciclos funcionales, definiendo el orden en que las pruebas van a ser ejecutadas. Para cada concepto del dominio del negocio, se prueba el ciclo de vida de los datos: crearlo, consultarlo, modificarlo y borrarlo [PTV02].

Dado que es muy común que existan cambios tardíos en los requerimientos, el proceso de prueba debe funcionar bien con esto, las siguientes son buenas prácticas a tener en cuenta: [KBP01]

- En lugar de desarrollar un gran conjunto de pruebas, desarrollar las pruebas a medida que se necesiten. Si los cambios al producto dejan obsoletas esas pruebas, al menos fueron útiles por un tiempo.
- No crear documentos de prueba muy grandes ya que tienen alto costo de mantenimiento.
- No basar las pruebas en los detalles de Interfaz de usuario pueden cambiar
- Diseñar pruebas automatizadas que maximicen la mantenibilidad y la portabilidad.
- Construir un conjunto de pruebas genéricas que trate situaciones que ocurren de producto en producto. Esto quita tiempo de planificación y facilita en el momento en que una nueva funcionalidad es agregada o es cambiada en forma tardía en el proyecto.
- Desarrollar un modelo de los usuarios del sistema y de los beneficios que ellos obtienen del sistema y derivar pruebas complejas de este modelo. Este modelo no cambia tanto a medida que el proyecto avanza ya que mira los beneficios en lugar de los detalles de implementación.

D2 – Validación de los Casos de Prueba

Objetivo

Asegurarse de que los casos de prueba definidos por el equipo de prueba son de valor para el negocio, por lo cual, debe ser validado por el cliente. En la Figura 34 se muestran los artefactos y roles involucrados en esta actividad.

Figura 34– D2 Validación de los Casos de Prueba

Descripción

Los casos de prueba definidos deben ser validados por el cliente, con el fin de que los casos de prueba y los datos de prueba sean de valor para el negocio. El orden y la importancia con que se validan los casos de prueba esta dada por las prioridades definidas en IP – Inventario de Prueba.

D3 – Asignación de los casos de prueba

Objetivo

Asignar los casos de prueba a los testers, definiendo el plan de ejecución del ciclo. En la Figura 35 se muestran los artefactos y roles involucrados en esta actividad.

Figura 35– D3 Asignación de los Casos de Prueba

Descripción

A partir de la planificación realizada para el ciclo, cada diseñador asigna a los testers a su cargo los casos de prueba a ejecutar, la asignación realizada se refleja en PEC- Plan de Ejecución del Ciclo. Una posible estrategia a considerar es la de rotación de los testers en los diferentes ciclos. [KBP01] recomienda esta estrategia ya que esto ayuda a que no se aburran, o se especialicen en algo y pierdan la generalidad. Si existe un único especialista en un área y éste se va, queda un agujero de conocimiento en el equipo. Otra forma de mitigar el riesgo de la especialización, es probar por pares. En general, el momento de realizar

la rotación es cuando el tester se siente conforme con el nivel de calidad alcanzado en las pruebas de su parte.

5.4.4 Ejecución de las Pruebas

Esta sub-actividad fue descrita en el Capítulo 4. A continuación se describen las actividades de la sub-etapa Ejecución de las Pruebas.

E1 - Pruebas de Humo

Objetivo

Validar que las funcionalidades esenciales del producto están presentes y básicamente funcionan. En la Figura 36 se muestran los artefactos y roles involucrados en esta actividad.

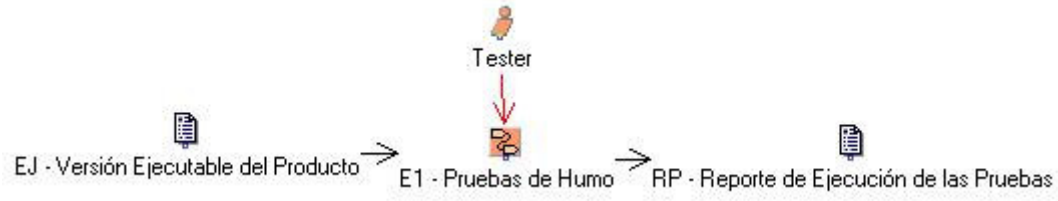

Figura 36 – E1 Pruebas de Humo

Descripción

Las pruebas de humo son un conjunto de pruebas aplicadas a cada nueva versión, su objetivo es validar que las funcionalidades básicas de la versión se cumplen según lo especificado. Estas pruebas buscan grandes inestabilidades o elementos clave faltantes o defectuosos, que hace imposible realizar las pruebas como fueron planificadas para el ciclo. Si la versión no pasa las pruebas de humo, no se comienza la ejecución de las pruebas de la versión [KBP01].

Las razones técnicas para rechazar una versión pueden ser [KBP01]:

- Si la importancia de la versión es que agrega una funcionalidad crítica y se descubre que no está presente en la versión o que falla inmediatamente
- Si las funcionalidades claves del producto no funcionan, la versión probablemente fue realizada con los archivos incorrectos.
- Si se recibe una versión pero se sabe que en poco tiempo se va a recibir otra, dependiendo del costo de instalación de la versión, seguramente convenga seguir probando la versión anterior hasta que llegue la nueva.

E2 - Ejecución de las pruebas

Objetivo

Ejecutar los casos de prueba planificados para el ciclo y observar su resultado. En la Figura 37 se muestran los artefactos y roles involucrados en esta actividad.

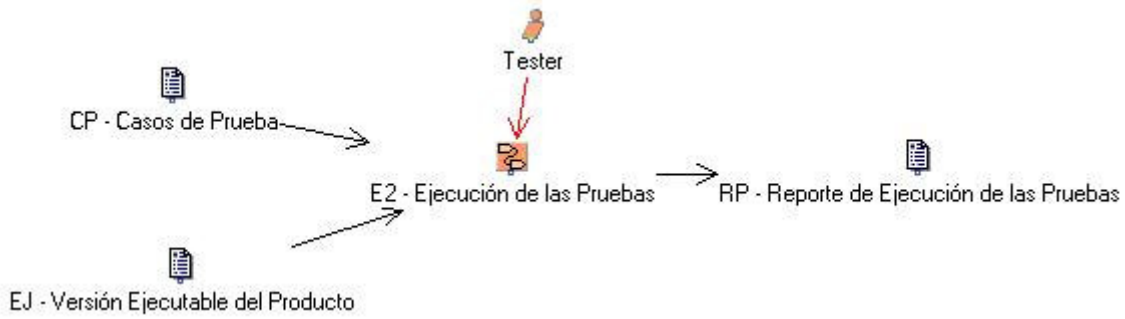

Figura 37- E2 Ejecución de las Pruebas

Descripción

Esta actividad incluye seleccionar los casos de prueba, ejecutarlos, registrar resultados y determinar si los incidentes encontrados son causados por errores en el producto o errores en las pruebas [Kit95]. En caso de encontrar incidentes, se realiza la actividad E4 – Reporte de Incidentes. La ejecución de las pruebas puede ser manual o automatizada. Cuando se ejecuta un caso de prueba y el resultado no se corresponde con el real, el tester debe:

- Volver a ejecutar el caso de prueba, al menos una vez, para asegurarse que se trata de un incidente.
- Ejercitar otros escenarios y otros datos, para poder aislar el incidente.

E3 - Testing Exploratorio

Objetivo

Mitigar la posibilidad de equivocase al realizar el análisis de riesgo del producto y estar dejando de lado funcionalidades importantes para el negocio. En caso de que en una sesión de testing exploratorio encuentre un defecto crítico para el negocio que no fue encontrado por las pruebas planificadas para el ciclo, se debe revisar el análisis de riesgo realizado. En la Figura 38 se muestran los artefactos y roles involucrados en esta actividad.

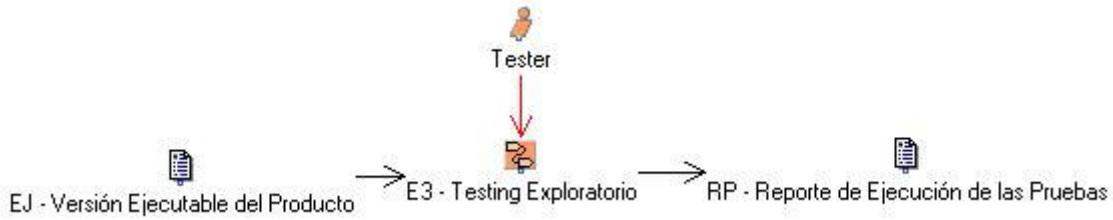

Figura 38 – E3 Testing Exploratorio

Descripción

El testing exploratorio se define como el aprendizaje, diseño y ejecución de las pruebas en forma simultánea [Bac01]. Los testers diseñan, desarrollan y ejecutan las pruebas durante la ejecución del producto. Explorar el producto permite verlo desde otra óptica, tiene la ventaja de que es barato y rápido.

Si se encuentran defectos durante el Testing Exploratorio que no fueron encontrados por las pruebas planificadas y que son críticos para el funcionamiento del producto o para el cliente, entonces, se debe revisar el análisis de riesgo realizado.

E4 - Reporte de incidentes

Objetivo

Reportar los incidentes encontrados durante las pruebas. En la Figura 39 se muestran los artefactos y roles involucrados en esta actividad.

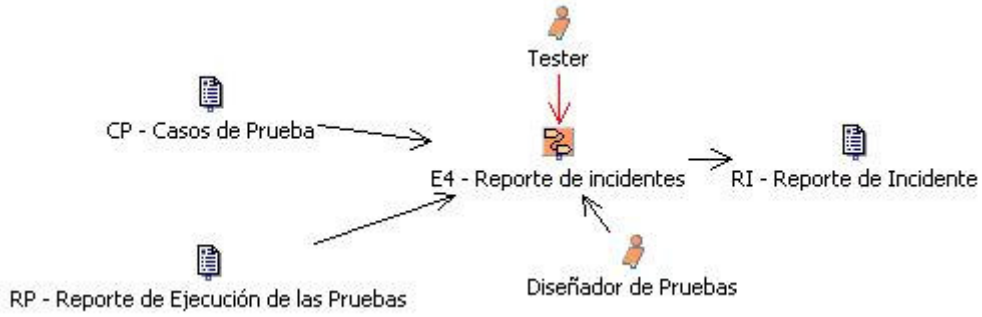

Figura 39– E4 Reporte de Incidentes

Descripción

Los incidentes se reportan siguiendo el proceso que se definió junto al Cliente en la actividad P8-Definición del Proceso de Incidentes.

Al reportar un incidente debe quedar claro si es un defecto, una mejora o una observación. Cada incidente reportado debe hacer referencia al caso de prueba que lo originó, debe poder reproducirse y estar acompañado de una valorización del impacto del mismo. El reporte del incidente debe ser preciso, simple y claro.

A continuación se listan algunos puntos básicos para el reporte de incidentes [KBP01]:

- Hacer la descripción de los pasos de reproducción en forma simple.
- Realizar los pasos para llegar al incidente de uno a la vez.
- Numerar cada paso.
- No saltearse ningún paso que sea necesario para la reproducción.
- Listar el conjunto más corto de pasos que llevan al lector a la falla.
- Usar sentencias cortas y simples.
- Indicar qué sucedió y que se esperaba que sucediera.
- Si las consecuencias son serias, explicar porque se piensa que lo son.
- Mantener un tono de redacción neutral.
- No realizar bromas, pueden generar malos entendidos.

E5 - Validación de los incidentes

Objetivo

El cliente valida los incidentes encontrados. En la Figura 40 se muestran los artefactos y roles involucrados en esta actividad.

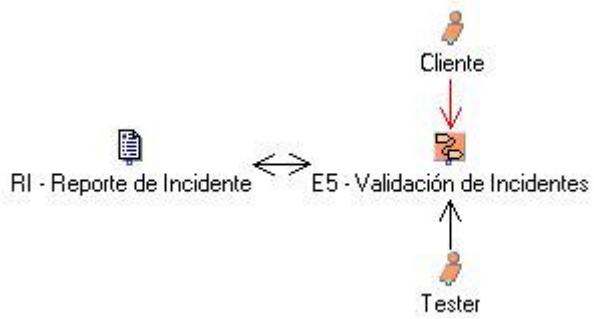

Figura 40 – E5 Validación de los Incidentes

Descripción

Los incidentes reportados por el equipo de prueba deben ser validados por el cliente, siguiendo el procedimiento definido en la actividad P8- Definición del Proceso de Incidentes. El cliente además de validar que realmente se trate de un incidente, puede cambiar la prioridad, el estado y la categoría del incidente.

E6 -Verificación de las Correcciones

Objetivo

Validar las correcciones realizadas por los desarrolladores de los incidentes encontrados, mediante la ejecución de pruebas de regresión. En la Figura 41 se muestran los artefactos y roles involucrados en esta actividad.

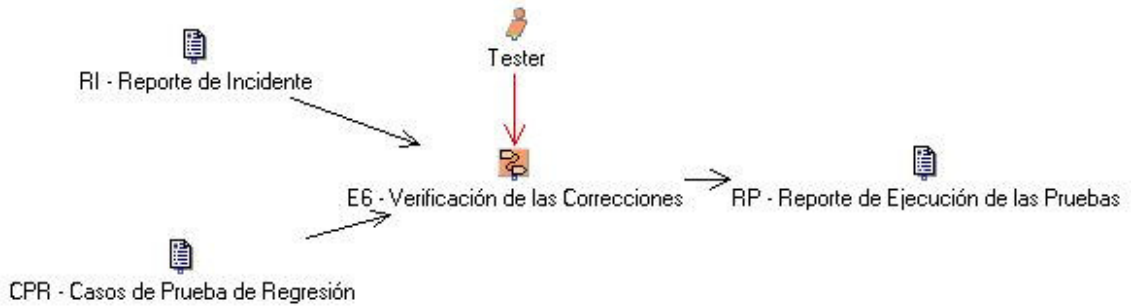

Figura 41 – E6 Verificación de las Correcciones

Descripción

Luego de terminada la ejecución de las pruebas para un ciclo de prueba, los incidentes reportados serán corregidos por los desarrolladores. Esta actividad puede ser realizada a partir del segundo ciclo de prueba, ya que revisa que el incidente se haya corregido en la nueva versión y que puede ser cerrado. Se ejecutan las pruebas de regresión para asegurar que los cambios no introducen un comportamiento no deseado o nuevos errores, esto implica seleccionar los casos de prueba a reejecutar y ejecutarlos. En caso de encontrar nuevos incidentes se reportan en la actividad E4-Reporte de Incidentes.

5.5 Evaluación del Proyecto

Esta etapa fue descrita en el Capítulo 4. A continuación se describen las actividades de la etapa Evaluación del Proyecto.

V1 - Evaluación de la satisfacción del Cliente

Objetivo

Esta actividad tiene como objetivo evaluar la satisfacción del cliente con el proyecto de prueba y el resultado de las pruebas. En la Figura 42 se muestran los artefactos y roles involucrados en esta actividad.

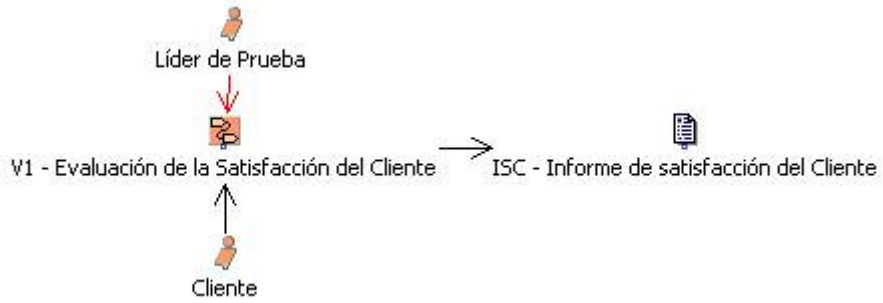

Figura 42– V1 Evaluación de la satisfacción del Cliente

Descripción

Al finalizar el proyecto, se debe conocer el grado de satisfacción del cliente respecto a los resultados obtenidos, la comunicación que se tuvo durante el proyecto y a su percepción del avance del proyecto. Estos elementos ayudan a ajustar y mejorar el proceso de prueba. La evaluación de la satisfacción se puede realizar en una entrevista o con un cuestionario.

V2 - Ajustes y Mejoras del Proceso de Prueba

Objetivo

Definir los ajustes y las mejoras necesarias en el proceso de prueba. En la Figura 43 se muestran los artefactos y roles involucrados en esta actividad.

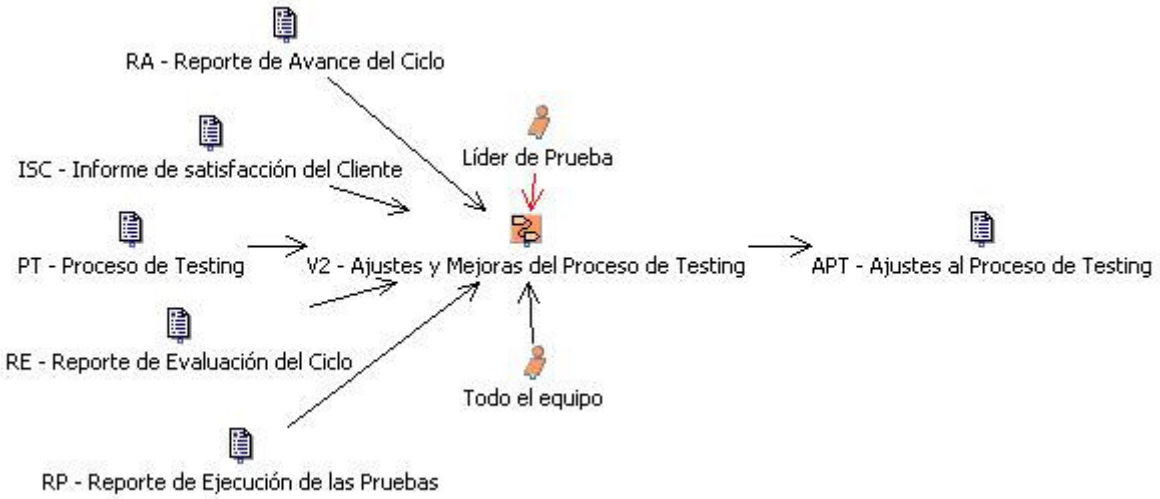

Figura 43- V2 Ajustes y Mejoras del Proceso de Prueba

Descripción

Dentro de los objetivos de la mejora del proceso están enriquecer la predicción y la calidad de lo entregado, mientras se mantiene (o se mejora) la productividad [PSP].

Los modelos de mejora del proceso de prueba como el Testing Maturity Model (TMM) [BS196], muestran la importancia de tener un grupo dedicado a la mejora del proceso de prueba, que aplique técnicas de mejora incremental del proceso y evalúe su impacto. Al finalizar el proyecto de prueba, el equipo de mejora del proceso de prueba junto con el equipo de prueba debe evaluar los procedimientos existentes para futuras experiencias. Según la naturaleza del dominio del negocio de cada producto de software a probar, pueden surgir procedimientos específicos, que es interesante recolectar y escribir, de forma de ayudar a la mejora del proceso y de guardar una memoria colectiva.

Los datos históricos se analizan antes de implementar cualquier cambio en el proceso, ayudan además a identificar las causas de los problemas [PSP].

Se deben implementar primero las mejoras de rentabilidad más alta. Aquellas que ofrecen la mayor mejora potencial en comparación al esfuerzo requerido para realizar los cambios. Después de hacer los cambios, supervisar los resultados del funcionamiento para determinar si las mejoras al proceso puestas en ejecución han sido eficaces [PSP].

En este trabajo se utiliza para la documentación del proceso y los cambios y mejoras en el mismo, la herramienta DeVeloPro, la cual se describe en el Anexo A. Se trata de una herramienta para la gestión de los procesos de una organización, permite definir los distintos elementos de un proceso: actividades, roles, artefactos, etapas, definiciones y herramientas. La definición de las actividades se realiza en forma gráfica. La herramienta permite versionar estos elementos de forma de conocer qué cambio, por qué cambio y quién lo cambio.

V3 - Reporte Final del Proyecto de Prueba

Objetivo

Realizar el informe final del proyecto de prueba. En la Figura 44 se muestran los artefactos y roles involucrados en esta actividad.

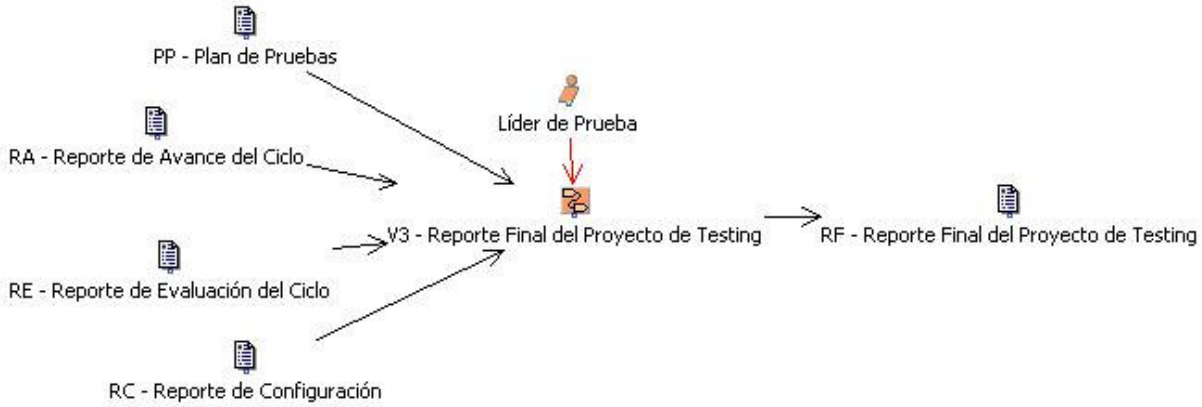

Figura 44– V3 Reporte Final del Proyecto de Prueba

Descripción

Una vez finalizado el proyecto de prueba, se realiza un reporte que resume el proyecto en su conjunto. Este reporte debe indicar el esfuerzo total, el esfuerzo por ciclo, por etapa y por actividad, las desviaciones que ocurrieron respecto a lo planificado y las razones de dichas desviaciones. Las mediciones realizadas durante este proyecto de prueba son la base para las estimaciones de los proyectos posteriores.

V4 - Archivo del Testware

Objetivo

Archivar el testware para su uso en proyectos posteriores. En la Figura 45 se muestran los artefactos y roles involucrados en esta actividad.

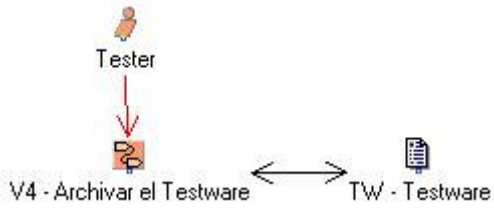

Figura 45 – V4 Archivo del Testware

Descripción

Esta actividad organiza los elementos que componen el testware de forma que puedan ser reutilizados en proyectos futuros. El reporte final de proyecto es parte del testware.

5.6 Actividades Comunes a las Etapas

Las siguientes actividades se realizan en todas las etapas de ProTest. Son las actividades que tienen que ver con la estimación del esfuerzo de cada actividad a realizar y su registro una vez que la actividad se realizó.
A partir de esa información se puede estudiar la desviación entre lo planificado y lo real, de forma de anticipar si es posible realizar las pruebas agendadas para el ciclo o si es necesario hacer una nueva

planificación del ciclo o de todo el proyecto. Permite conocer además cuales son los casos de prueba que ocupan el mayor esfuerzo.

G1 - Estimación de Tareas

Objetivo

Cada integrante del equipo de pruebas debe estimar el tiempo que le insumirá cada actividad asignada en el ciclo. En la Figura 46 se muestran los artefactos y roles involucrados en esta actividad.

Figura 46- G1 Estimación de Tareas

Descripción

Se debe desarrollar la planificación del trabajo antes de comenzarlo, basándose en las actividades del proceso definido y los datos históricos personales [PMC05]. La persona que realiza la tarea es quien debe estimar el tiempo que le llevará realizarla [KBP01].

Esta actividad se realiza en conjunto con la actividad S1-Planificación del Ciclo, donde las estimaciones de cada miembro del equipo son organizadas en el tiempo.

La estimación es un proceso de aprendizaje, se mejora con la experiencia y los datos históricos. Las estimaciones están sujetas a error, es mejor estimar por partes, ya que el error total debe ser menor que la suma de los errores de las partes si las partes son estimadas independientemente. Se usan datos históricos para hacer las estimaciones [PMC05].

G2 - Reporte de Esfuerzo

Objetivo

Mantener un registro del esfuerzo que insume realizar cada tarea en el proyecto de prueba. En la Figura 47 se muestran los artefactos y roles involucrados en esta actividad.

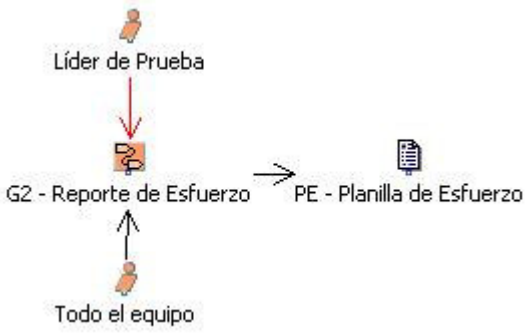

Figura 47– G2 Reporte de Esfuerzo

Descripción

Cada integrante del equipo de pruebas debe registrar el tiempo real (horas/persona) que le insumió realizar cada actividad de ProTest en el proyecto de prueba.

Capítulo 6

ARTEFACTOS Y ROLES DE PROTEST

Para poder realizar una actividad, deben estar presentes sus artefactos de entrada. Los participantes utilizan esos artefactos para realizar las tareas definidas en la actividad y generar los artefactos de salida. Existen artefactos que son consultados en una actividad, esos son los artefactos de entrada. Otros, son creados en la actividad, esos son los artefactos de salida. Los artefactos que son modificados por la actividad son aquellos que aparecen como entrada y salida de la misma.

Un rol define el comportamiento y las responsabilidades de una persona ó un grupo de personas trabajando en equipo. Una persona puede tener varios roles y un mismo rol puede ser cumplido por más de una persona. ProTest define los siguientes roles para el equipo de prueba: Líder de Prueba, Diseñador de Pruebas y Tester. Existen dos roles externos al equipo de prueba: el Cliente y el Desarrollador o Contraparte Técnica.

En este capítulo se describen los Artefactos y los Roles de ProTest. En la sección 6.1 se muestra la lista de artefactos y se describe el contenido de cada uno. En la sección 6.2 se describen los roles del proceso, para cada rol se muestran los requisitos y competencias, las actividades en las que participa y las responsabilidades en cada actividad.

6.1 Artefactos

En la Tabla 11 se listan los artefactos existentes en el proceso, para cada uno se muestra el rol responsable de generarlo, las actividades donde es entrada y las actividades que modifican el artefacto, donde el mismo es salida.

Artefacto	Responsable	Entrada en Actividades	Salida en Actividades
ACP - Agenda de Ciclos de Prueba	Líder de Prueba	P7 – Planificación de las Pruebas P5 – Definición de los Ciclos de Prueba	P1 - Negociación con el Cliente P5 – Definición de los Ciclos de Prueba
APT - Ajustes al Proceso de Prueba	Líder de Prueba		V2 – Ajustes y Mejoras del Proceso de Prueba
CP - Casos de Prueba	Diseñador de Pruebas	S3 – Seguimiento y Control del Ciclo S4 – Evaluación de las Pruebas D2 – Validación de los Casos de Prueba D3 – Asignación de los Casos de Prueba E2 – Ejecución de las Pruebas E4 – Reporte de Incidentes	D1 – Diseño de los Casos de Prueba
CPR - Casos de Prueba de Regresión	Diseñador de Pruebas	E6 – Verificación de las Correcciones	
EJ - Versión Ejecutable del Producto	Tester	P4 – Exploración del Producto E1 – Pruebas de Humo E2 – Ejecución de las Pruebas E3 – Testing Exploratorio	C2 – Instalación y Configuración
ET - Estimación de Tareas	Líder de Prueba	S1 – Planificación del Ciclo	G1 – Estimación de Tareas
FR - Fuentes de Requerimientos	Cliente	I2 – Revisión Preliminar de requerimientos P2 - Revisión de requerimientos	
IP - Inventario	Diseñador de Pruebas	I1 – Definición del Servicio I2 – Revisión Preliminar de requerimientos I3 - Análisis Preliminar de Riesgo del Producto P1 – Negociación con el Cliente P2 – Revisión de Requerimientos P3 – Análisis de Riesgo del Producto P4 – Exploración del Producto P5 – Definición de los Ciclos de Prueba P7 – Planificación de las Pruebas S1 – Planificación del Ciclo D1 – Diseño de los Casos de Prueba P6 – Definición del Testware	I1 – Definición del Servicio I2 – Revisión Preliminar de requerimientos I3 - Análisis Preliminar de Riesgo del Producto P1 – Negociación con el Cliente P2 – Revisión de Requerimientos P3 – Análisis de Riesgo del Producto P4 – Exploración del Producto

Artefacto	Responsable	Entrada en Actividades	Salida en Actividades
ISC- Informe de Satisfacción del Cliente	Líder de Prueba	V2 – Ajustes y Mejoras del Proceso de Prueba	V1- Evaluación de la Satisfacción del Cliente
MT - Matriz de Trazabilidad	Diseñador de Pruebas	S3 – Seguimiento y Control del Ciclo	D1 – Diseño de los Casos de Prueba
PD - Plan de Desarrollo del Producto	Cliente	I1 – Definición del Servicio P1 - Negociación con el Cliente P6 – Definición del Testware P5 – Definición de los Ciclos de Prueba	
PE - Planilla de Esfuerzo	Líder de Prueba	S3 – Seguimiento y Control del Ciclo G2 – Reporte de Esfuerzo	G2 – Reporte de Esfuerzo
PEC - Plan de Ejecución del Ciclo	Diseñador de Pruebas		D3 – Asignación de los Casos de Prueba
PI - Proceso de Incidentes	Líder de Prueba	P7 – Planificación de las Pruebas	P8 – Definición del Proceso de Incidentes
PP - Plan de Pruebas	Líder de Prueba	S1 – Planificación del Ciclo G1 – Estimación de Tareas S4 – Evaluación de las Pruebas V3 – Reporte Final del Proyecto	P7 – Planificación de las Pruebas
PPC - Plan Pruebas del Ciclo	Líder de Prueba	S3 – Seguimiento y Control del Ciclo	S1 – Planificación del Ciclo
PS – Propuesta de Servicio	Líder de Prueba		I1 – Definición del Servicio
PT - Proceso de Prueba	Líder de Prueba	V2 – Ajustes y Mejoras del Proceso de Prueba	
RA - Reporte de Avance del Ciclo	Líder de Prueba	S1 – Planificación del Ciclo G1 – Estimación de Tareas V2 – Ajustes y Mejoras del Proceso de Prueba V3 – Reporte Final del Proyecto	S3 – Seguimiento y Control del Ciclo
RC - Reporte de Configuración	Tester	V3 – Reporte Final del Proyecto	S2 – Administración de la Configuración
RE - Reporte de Evaluación del Ciclo	Líder de Prueba	V2 – Ajustes y Mejoras del Proceso de Prueba V3 – Reporte Final del Proyecto	S4 – Evaluación de las Pruebas
RF - Reporte Final del Proyecto de Prueba	Líder de Prueba		V3 – Reporte Final del Proyecto
RI - Reporte de Incidente	Tester	S4 – Evaluación de las Pruebas E5 – Validación de los Incidentes E6 – Verificación de las Correcciones S3 – Seguimiento y Control del Ciclo	E4 – Reporte de Incidentes E5 – Validación de los Incidentes
RO - Reporte de Obstáculos	Tester	S4 – Evaluación de las Pruebas	S3 – Seguimiento y Control del Ciclo C1 – Instalación de herramientas C2 – Instalación y Configuración

Artefacto	Responsable	Entrada en Actividades	Salida en Actividades
RP - Reporte de Ejecución de las Pruebas	Tester	S3 – Seguimiento y Control del Ciclo S4 – Evaluación de las Pruebas E4 – Reporte de Incidentes V2 – Ajustes y Mejoras del Proceso de Prueba	E1 – Pruebas de Humo E2 – Ejecución de las Pruebas E3 – Testing Exploratorio E6 – Verificación de las Correcciones
RR - Resumen de Reunión	Líder de Prueba		I1 – Definición del Servicio P1 - Negociación con el Cliente D2 – Validación de los Casos de Prueba
TW - Testware	Diseñador de Pruebas	C1 –Instalación de Herramientas V4 – Archivar el Testware	P6 – Definición del Testware V4 – Archivar el Testware

Tabla 11 – Artefactos y sus relaciones

A continuación se describe cada uno de los artefactos de ProTest.

Agenda de Ciclos de Prueba (ACP)

En la Agenda de Ciclos de Prueba se definen los ciclos de prueba que se realizarán durante el proyecto de prueba, las fechas tentativas de comienzo y fin de cada ciclo, la relación con las versiones de desarrollo y las lista de funcionalidades que serán probadas en cada ciclo, incluyendo la prioridad de cada funcionalidad. Se corresponde con el punto 12 del artefacto PP- Plan de Pruebas.

Ajustes al Proceso de Prueba (APT)

Contiene los cambios y mejoras a realizar al Proceso de Prueba, obtenidos como resultado de haberlo utilizado en un proyecto particular. Los ajustes deben indicar qué debe cambiar, porque debe ser cambiado y quien propone el cambio.

Cuando se decide realizar un cambio en ProTest, dicho cambio se realiza con la herramienta DeVeloPro, que permite versionar estos elementos de forma de conocer qué se cambio, por qué se realizó el cambio y quién lo hizo. DeVeloPro se describe en el Anexo A de este trabajo.

Caso de Prueba (CP)

Cada caso de prueba consiste de tres valores: acciones, datos y resultados esperados [Bla02].

Los datos necesarios para identificar cada caso de prueba incluyen:

- **Identificador**: Identificador único para el caso de prueba
- **Conjunto de prueba**: Identificador de los conjuntos de prueba donde el caso de prueba es usado
- **Nombre**
- **Funcionalidad**: Funcionalidad del IP-Inventario de Prueba que se prueba con este caso de prueba.
- **Prioridad**: Prioridad asignada al caso de prueba
- **HW**: Lista de Hardware requerido para ejecutar el caso de prueba
- **SW**: Lista de Software requerido para ejecutar el caso de prueba
- **Configuración** (setup): Lista de pasos necesarios para comenzar las pruebas

- **Retroceder acciones** (cleanup): Lista de pasos necesarios para retroceder al estado previo a la prueba
- **Condiciones**: Lista de condiciones que deben cumplirse al ejecutar las pruebas
- **Resultado esperado**: Especificación del resultado esperado de ejecutar las pruebas

Los siguientes datos se completan una vez que el caso de prueba se ejecutó:

- **Resultado real**: Especificación de lo que realmente ocurrió: Pasó/Falló/No se pudo ejecutar
- **Duración**: Tiempo reloj en ejecutar el caso de prueba
- **Esfuerzo**: Horas-Persona requeridas para configurar y ejecutar el caso de prueba
- **Fecha**: Fecha en la que se realizó la prueba
- **Identificación del Defecto**: En caso de que la prueba falle se debe identificar el incidente reportado para su seguimiento
- **Versión**: Identificador de la versión probada
- **Tester**: Nombre del tester que ejecutó el caso de prueba

Casos de Prueba de Regresión (CPR)

Los casos de prueba de regresión son los casos de prueba que se define ejecutar al comenzar cada ciclo, para verificar que un incidente fue corregido y puede ser cerrado.

Se mantienen los datos de identificación del caso de prueba y se modifican los datos referentes a la ejecución, reflejando el resultado de ejecutar el caso de prueba para la nueva versión del producto.

Versión Ejecutable del Producto (EJ)

Ejecutable de la versión a probar del producto para cada ciclo de prueba, debe ser identificada de forma única, estar acompañada de las notas de la versión donde se explican los cambios y mejoras incorporados a la versión.

Estimación de Tareas (ET)

Contiene el esfuerzo estimado para cada persona de las actividades planificadas en el ciclo. La granularidad de estas estimaciones es semanal.

Fuentes de Requerimientos (FR)

Son las fuentes desde donde se obtienen las especificaciones del producto. Éstas pueden ser documentos de requerimientos, sistemas ya existentes, manuales del sistema, reportes de defectos, prototipos y entrevistas con cliente y usuarios.

Informe de Satisfacción del Cliente (ISC)

Resumen de la satisfacción del cliente respecto a los resultados obtenidos y el trabajo realizado por el equipo de prueba.

Inventario de Prueba (IP)

El inventario es una lista de las funcionalidades que serán probadas. Para cada funcionalidad se asigna:

- **Identificador**: Referencia única a la especificación de requerimientos, donde se encuentra la descripción detallada del mismo.
- **Nombre**: Nombre de la funcionalidad a probar.
- **Complejidad**: Indica lo complejo que es para los desarrolladores implementar dicha funcionalidad. Los valores posibles son: Alta, Media y Baja.
- **Criticidad**: Indica cuán crítica para el negocio es dicha funcionalidad. Los valores posibles son: Alta, Media y Baja.
- **Prioridad**: Indica la prioridad que tiene esa funcionalidad para las pruebas. Los valores posibles son: Alta, Media y Baja.

Matriz de Trazabilidad (MT)

Para cada elemento del IP - Inventario de Prueba, se crea una matriz que muestra la correspondencia entre la funcionalidad a probar y los casos de prueba definidos para probar dicha funcionalidad. Esto nos permite saber qué casos de prueba cubren qué requerimientos de prueba [Kit95].

La trazabilidad de las funcionalidades a los casos de prueba, permite realizar un análisis de impacto si cambian los requerimientos.

Plan de Desarrollo del Producto (PD)

Plan donde se especifica el cronograma de las distintas versiones del producto que serán generadas y las funcionalidades incluidas en cada versión. Este plan es confeccionado por el equipo de desarrollo y es utilizado por el equipo de prueba para poder planificar las pruebas de cada versión.

Planilla de Esfuerzo (PE)

Planilla donde se registran el esfuerzo (horas/persona) que insumió cada actividad del proyecto de prueba. Se reportan:

- **Actividad**: Identificador de la actividad que se reporta
- **Fecha**: Día en que se realizó la actividad
- **Esfuerzo**: Minutos que se realizó la actividad
- **Observaciones**: Descripción breve de la actividad realizada

Existen actividades sobre las que interesa conocer más información. Para las siguientes actividades, interesa contar con el tiempo que insume como se detalla en la Tabla 12.

Actividad	Información extra
P2 –Revisión de Requerimientos	Tiempo en revisar cada requerimiento
D1 – Diseño de los Casos de Prueba	Tiempo por caso de prueba
E2 – Ejecución de las Pruebas	Tiempo que lleva: Ejecutar el caso de prueba la primera vez Ejecutar el caso de prueba para verificar que se encontró un incidente. Explorar otros casos de prueba para ver las condiciones en las cuales el incidente se manifiesta
E3 – Testing Exploratorio	Tiempo por sesión de testing exploratorio
E4 – Reporte de Incidentes	Tiempo por incidente
E6 – Verificación de las correcciones	Tiempo que lleva ejecutar cada caso de prueba de regresión

Tabla 12 – Información extra para Reporte de Actividades

Plan de Ejecución del Ciclo (PEC)

En el Plan de Ejecución del Ciclo se registran los casos de prueba, el orden en que serán ejecutados en el ciclo y quienes son los testers encargados de ejecutarlos.

Proceso de Incidentes (PI)

El proceso de Incidentes registra los estados por los que pasa un incidente, desde que es descubierto por el equipo de pruebas hasta que es reparado por el equipo de desarrollo. Este artefacto es parte del plan de pruebas, corresponde al punto 9 del PP – Plan de Prueba. Contiene:

- Estados del proceso de incidentes
- Personas involucradas en el proceso de incidentes
- Condiciones para pasar de un estado a otro del proceso de incidentes

Plan de Pruebas (PP)

En el Plan de Pruebas se define quién, cuándo, dónde y cómo se realizarán las actividades en el proyecto de prueba. Los puntos principales del Plan de Pruebas son [Bla02]:

1. Introducción
2. Agenda
3. Condiciones
4. Requerimientos del Entorno
5. Desarrollo de las pruebas
6. Ejecución de las pruebas
7. Personas y roles
8. Requerimientos de Capacitación
9. Proceso de Seguimiento de Incidentes
10. Clasificación de los defectos
11. Administración de las Versiones
12. Ciclos
13. Entregables

A continuación se describe brevemente cada uno de estos puntos:

1. **Introducción:** Identifica qué va a ser probado, resume el enfoque general usado para las pruebas, los objetivos y metodologías usados. Contiene:
   - **Alcance**: Describe que cosas están dentro del esfuerzo de prueba y que cosas no. Una forma de presentarlo es mediante una tabla donde se indiquen que funcionalidades están y cuales no están dentro del esfuerzo de prueba.
   - **Lugar**: Especifica el lugar físico donde se realizarán las pruebas.
   - **Definiciones**: En esta sección se incluye un glosario de términos para clarificar la terminología para quienes no tienen experiencia en el campo de las pruebas.
   - **Referencias**: Listar los documentos referenciados por este plan junto con su versión.

2. **Agenda**: Se presenta la agenda del proyecto de prueba, con fecha de comienzo y fin de cada tarea. Contiene comienzo y fin de cada una de las etapas y sub-etapas del proceso de prueba.

3. **Condiciones:** Describe las condiciones esenciales que deben cumplirse para poder realizar cada etapa de las pruebas y para poder continuar con un proceso de pruebas efectivo. Contiene:
   - **Condiciones de entrada**: Se deben cumplir para comenzar las pruebas de cada ciclo del sistema, por ejemplo:
     - El sistema de seguimiento de incidentes fue definido.
     - Todos los componentes están bajo administración de la configuración.
     - El equipo de desarrollo configuró el entorno. El equipo de pruebas tiene los permisos adecuados de acceso al sistema.
     - El equipo de desarrollo ha completado todas las funcionalidades (y reparado todos los defectos) establecidas para esta versión.
     - El equipo de desarrollo ha realizado las pruebas unitarias de todos los componentes de esta versión.
     - El equipo de desarrollo provee el software al equipo de pruebas 3 días calendario antes de comenzar las pruebas.
     - El equipo de prueba realiza pruebas de humo durante 3 días y reporta los resultados en una reunión junto con el gerente del proyecto. En esa reunión deben resolverse los plazos en caso de no poder seguir las pruebas.
   - **Condiciones para continuar**: Se deben cumplir para continuar las pruebas del ciclo de un sistema, por ejemplo:
     - Todo el software liberado al equipo de pruebas está acompañado de notas de la versión.
     - Se hacen reuniones semanales de revisión de incidentes para gestionar los incidentes abiertos y los tiempos de cierre.
   - **Condiciones de salida**: Se deben cumplir para terminar las pruebas de un ciclo, por ejemplo:
     - El equipo de pruebas ha ejecutado todas las pruebas planificadas.
     - El equipo de pruebas ha verificado que todos los incidentes en el sistema de seguimiento están cerrados o rechazados, verificado por medio de pruebas de regresión.
     - Se tiene una reunión entre el Gerente del Proyecto y el equipo de prueba para acordar que se ha terminando el ciclo.

4. **Requerimientos del Entorno**: Presenta los recursos no humanos que se requieren para las pruebas.
   - **Hardware Base**: Presenta los requerimientos de Hardware necesarios para probar el sistema.
   - **Software Base**: Presenta los requerimientos de Software necesarios para probar el sistema.
   - **Herramientas**: Presenta las herramientas necesarias para probar el sistema.

5. **Desarrollo de las pruebas**: Describe la forma en que el equipo de pruebas va a crear el Testware y cómo se generarán los datos para las pruebas.

6. **Ejecución de las pruebas**: Describe la forma en que serán manejadas las versiones, las pruebas y los datos durante la ejecución.

7. **Personas y Roles**: Se definen los roles involucrados en el proyecto de prueba, las responsabilidades y el nombre de las personas que realizaran dichos roles, junto con la información de contacto.

8. **Requerimientos de Capacitación**: Describe los requerimientos de capacitación necesarios para realizar las pruebas.

9. **Proceso de Seguimiento de Incidentes**: Describe el sistema usado para reportar y seguir los incidentes.

10. **Clasificación de los Incidentes**: Describe la forma en que se clasifican los incidentes encontrados.

11. **Administración de las Versiones**: Describe quién será el encargado de instalar y configurar las versiones en el ambiente de pruebas.

12. **Ciclos**: Este punto se corresponde con el artefacto: ACP-Agenda de los Ciclos de Prueba.

13. **Entregables**: Se especifican los artefactos que serán producidos durante el proyecto de prueba y serán entregados al cliente.

Plan Pruebas del Ciclo (PPC)

El Plan de Pruebas del Ciclo contiene la planificación de las actividades a realizar en un ciclo de pruebas, incluye:

- Agenda planificada para el ciclo, con las actividades a realizar por cada miembro del equipo durante el ciclo y las horas que dedicarán a cada actividad
- Funcionalidades que serán probadas en el ciclo y orden en que serán probadas
- Incidentes reparados en la versión
- Cambios realizados en la versión
- Pruebas de regresión que se ejecutarán en el ciclo
- Desviaciones respecto a lo planificado en PP-Plan de Pruebas
- Riesgos del proyecto de prueba y mecanismos para su mitigación y contingencia.

Proceso de Prueba (PT)

Es el proceso seguido para probar el producto, en este caso es ProTest. Deben estar disponibles a todos los miembros de la organización, por lo cual se debe contar con documentación accesible del proceso, en un formato imprimible. Los cambios introducidos en el proceso requieren generar nuevamente la documentación. Si el proceso tiene gran número de actividades, cada cambio puede resultar trabajoso de realizar, pudiendo quedar inconsistente la información en distintos elementos del proceso.

Para la documentación de ProTest se utiliza la herramienta DeveloPro, que se describe en el Anexo A de este trabajo. DeVeloPro genera automáticamente el sitio web con páginas html estáticas y un documento en formato pdf, para cada versión del proceso definida en la herramienta.

Propuesta de Servicio (PS)

La propuesta de servicio es el resultado de la etapa Estudio Preliminar, resume el alcance del proyecto de prueba, la agenda preliminar de los ciclos de prueba y la cotización del mismo.

Reporte de Avance del Ciclo (RA)

El reporte de avance del ciclo es un resumen del trabajo realizado en el ciclo de prueba. Contiene:

- Las mediciones realizadas durante el ciclo
- Gráficas mostrando la información resumida
- La agenda real del ciclo y la desviación respecto a la agenda planificada
- Los obstáculos ocurridos durante el ciclo de prueba.
- Acciones tomadas al realizar el control del ciclo

Para poder reportar el avance del ciclo, se debe haber medido los siguientes elementos:

- **Funcionalidades**: Mediciones relacionadas con las funcionalidades probadas durante el ciclo.
- **Casos de prueba**: Mediciones relacionadas con la planificación, diseño y ejecución de los casos de prueba.
- **Incidentes**: Mediciones relacionadas con los incidentes encontrados durante el ciclo.
- **Esfuerzo**: Tiempo invertido en realizar cada actividad del Proceso.
- **Obstáculos**: Problemas que ocurrieron durante las pruebas

En la Tabla 13 se listan las mediciones que se tienen en cuenta en ProTest. Dichas mediciones fueron recolectadas de [ISQ05], [Kit95] y [Kan01].

Elemento	Métrica	Fórmula
Funcionalidad	Cubrimiento de Funcionalidades	Cantidad de Casos de prueba ÷ Total de funcionalidades a probar
	Tiempo por funcionalidad	Total de Tiempo en Ejecutar las Pruebas ÷ Total de funcionalidades probadas
	Densidad de defectos por funcionalidad	Total de Incidentes encontrados ÷ Total de funcionalidades probadas
	Cambios en las funcionalidades	Contar las funcionalidades que han cambiado desde la última versión
	Porcentaje de pruebas que cambiaron por cambios en las funcionalidades	Contar la cantidad de pruebas modificadas debido a cambios en funcionalidades ÷ Total de pruebas
Casos de prueba	Casos de prueba exitosos (1)	Contar cuantos casos de prueba fueron ejecutados y su resultado fué "Pasó"
	Casos de prueba que fallaron (2)	Contar cuantos casos de prueba fueron ejecutados y su resultado fué "Falló"
	Casos de prueba ejecutados	Es la suma (1) y (2) , también puede ser calculado como la suma de (3) y (4)
	Casos de prueba no ejecutados	Contar cuantos casos de prueba planificados para el ciclo no fueron ejecutados
	Casos de prueba planificados	Contar cuantos casos de prueba se planificaron realizar en el ciclo
	Casos de prueba disponibles	Contar cuantos casos de prueba fueron diseñados en el ciclo
	Casos de prueba disponibles ejecutados (3)	Contar cuantos de los casos de prueba disponibles en el ciclo, fueron ejecutados en el ciclo.

Elemento	Métrica	Fórmula
Casos de prueba (continuación)	Pruebas de regresión ejecutadas (4)	Contar la cantidad de casos de prueba ejecutados en el ciclo, que ya fueron ejecutados en ciclos anteriores.
	Tiempo medio por caso de prueba	Tiempo total en ejecutar los casos de prueba ÷ Total de casos de prueba ejecutados
	Densidad de incidentes por caso de prueba	Total de incidentes reportados en el ciclo ÷ Total de casos de prueba ejecutados
	Eficiencia de las pruebas	Cantidad de pruebas ejecutadas ÷ total de incidentes encontrados
Testing exploratorio	Sesiones de testing exploratorio	Cantidad de sesiones de testing exploratorio que se realizaron en el ciclo
	Esfuerzo de testing exploratorio	Contar el tiempo dedicado al testing exploratorio en el ciclo
	Incidentes encontrados en el testing exploratorio	Contar los incidentes encontrados durante el testing exploratorio del ciclo
	Funcionalidades exploradas	Contar las funcionalidades exploradas durante el testing exploratorio del ciclo
Incidentes	Incidentes por estado	Para cada estado (nuevo, resuelto, pendiente, reabierto, etc.) Contar los incidentes de ese estado en el ciclo
	Incidentes por categoría	Para cada categoría de incidentes definida, contar la cantidad de incidentes reportados en el ciclo
	Incidentes por tipo	Para cada tipo de incidente definido, contar la cantidad de incidentes reportados en el ciclo
	Incidentes por severidad	Para cada severidad de incidentes definida, contar la cantidad de incidentes reportados en el ciclo.
	Incidentes que no se pudieron reproducir	Contar los incidentes encontrados que no se pudieron reproducir por el equipo de desarrollo en el ciclo
	Falsos incidentes	Contar los incidentes reportados por el equipo de prueba en el ciclo que el cliente no consideró un incidente
	Tasa de reparación de incidentes	Cantidad de incidentes reparados para este ciclo ÷ Cantidad de incidentes encontrados en el ciclo anterior.
	Tiempo medio en resolver incidentes	Total de tiempo en resolver los incidentes desde que se reportaron ÷ Cantidad de incidentes resueltos en el ciclo
	Calidad de la reparación de incidentes	Cantidad de incidentes reabiertos en el ciclo ÷ Cantidad de incidentes reparados para el ciclo
	Tiempo medio en reportar incidentes	Total de tiempo en ejecutar casos de prueba ÷ Cantidad de incidentes reportados en el ciclo

Elemento	Métrica	Fórmula
Esfuerzo	Esfuerzo planificado (5)	Sumar el tiempo planificado para todas las actividades del proyecto de prueba
	Esfuerzo realizado en el ciclo	Sumar el tiempo invertido en todas las actividades del ciclo
	Esfuerzo realizado hasta el momento (6)	Sumar el tiempo realizado desde que el proyecto comenzó hasta el último ciclo realizado
	Esfuerzo por realizar (7)	Sumar el tiempo que se planifica para cada actividad hasta que el proyecto de prueba termine
	Esfuerzo total (8)	Sumar (6) y (7)
	Desviación en el esfuerzo planificado	Restar (8) menos (5)
	Esfuerzo por persona	Contar el esfuerzo por persona del equipo de prueba
	Esfuerzo en reportar incidentes	Sumar el tiempo en reportar los incidentes encontrados en el ciclo
	Esfuerzo por actividad	Para cada actividad del ciclo, sumar el tiempo en realizarla por cada integrante
	Esfuerzo por etapa	Para cada etapa del ciclo, sumar el tiempo en realizarla por cada integrante
Obstáculos	Obstáculos reportados	Contar la cantidad de problemas que fueron reportados en el ciclo
	Esfuerzo en obstáculos	Contar el tiempo invertido por el equipo de pruebas en resolver problemas
	Casos de prueba bloqueados	Contar la cantidad de pruebas bloqueadas por incidentes
	Pruebas de Humo que fallaron	Contar la cantidad de tentativas falladas de pasar las pruebas de humo en el ciclo
	Esfuerzo perdido en el ambiente de prueba	Sumar el esfuerzo invertido en problemas con el ambiente (por ejemplo dificultades para obtener el equipo de prueba o configurar el sistema)
	Incidentes debidos al ambiente de prueba	Sumar la cantidad de incidentes relacionados con el ambiente de prueba

Tabla 13 – Métricas sobre el Avance del Ciclo

Según las propiedades de calidad definidas para el producto a probar, la norma ISO 9126 define métricas externas que permiten evaluar calidad del producto de software durante la prueba. En la Tabla 2 de la sección 3.2.1 del Capítulo 3 se listan algunas de las métricas que podrían ser aplicadas al producto.

Reporte de Configuración (RC)

Reporte en el que se incluye el estado de los elementos de configuración para el proyecto de prueba. Este reporte muestra la línea base del proyecto de prueba, con las últimas versiones de cada elemento.

Reporte de Evaluación del Ciclo (RE)

Al finalizar el ciclo, se realiza un informe que evalúa las pruebas en el ciclo. Este informe resume la información incluida en el RA - Reporte de Avance del Ciclo y es entregado al cliente. Podría ser el mismo informe, pero muchas veces al cliente no le interesa conocer tanto detalle sobre el proyecto de prueba, como por ejemplo el esfuerzo invertido por el equipo de prueba en cada actividad. Se incluye una evaluación de la efectividad de las pruebas respecto al criterio de completitud.

Reporte Final del Proyecto de Prueba (RF)

Reporte que resume el proyecto en su conjunto. Este reporte totaliza la información recolectada en cada RA- Reporte de Avance del Ciclo. . Contiene:

- Las mediciones para todo el proyecto de prueba: Se realizan las mediciones de la Tabla 13 para todo el proyecto, también se realizan las mediciones de la Tabla 14.
- Gráficas mostrando la información resumida
- La agenda real del proyecto y la desviación respecto a la agenda planificada
- Los obstáculos ocurridos durante el proyecto de prueba
- Visión global de la calidad del producto

Métrica	Fórmula
Ciclos planificados	Cantidad de ciclos planificados para el proyecto de prueba
Ciclos realizados	Cantidad de ciclos de prueba realizados durante el proyecto
Desviación entre lo planificado y lo real	Diferencia entre el Esfuerzo planificado para el proyecto de pruebas y el Esfuerzo total real del proyecto.
Cambios al plan de pruebas	Cantidad de cambios realizados al plan de pruebas durante el proyecto
Porcentaje de artefactos entregados	Cantidad de artefactos entregados al cliente en el proyecto/Total de artefactos planificados a entregar
Incidentes sin resolver al terminar el proyecto de prueba	Cantidad de incidentes sin resolver al momento de terminar el proyecto de prueba
Tendencia de los incidentes en los ciclos	Mostrar la cantidad de incidentes encontrados en el tiempo

Tabla 14 – Métricas del Informe Final del Proyecto

Reporte de Incidente (RI)

Su propósito es documentar cualquier evento en la ejecución de las pruebas que requiera investigación posterior. Los elementos mínimos a documentar para cada incidente son:

- **Identificador**: Identificador único.
- **Caso de prueba**: Identificador del caso de prueba donde se encontró el incidente.
- **Resumen**: Descripción breve del incidente.
- **Pasos**: Pasos necesarios para la reproducción del incidente.
- **Versión**: Versión del producto que se está probando.
- **Prioridad**: Importancia del incidente encontrado, esta puede ser:
  - Crítica: La ejecución del sistema es interrumpida, no se puede seguir con la prueba
  - Alta: La aplicación sigue funcionado pero el problema tiene un alto impacto
  - Media: El problema tiene impacto medio en la aplicación
  - Bajo: El problema tiene poco impacto en la aplicación
- **Categoría**: Categoría a la que corresponde el incidente.

- **Tester**: Nombre del tester que reporta el incidente.
- **Estado**: Estado del incidente según el PI – Proceso de Incidentes definido.

Reporte de Ejecución de las Pruebas (RP)

El reporte de ejecución de las pruebas contiene la información referente a ejecutar cada CP-Caso de Prueba. Es la agrupación de los casos de prueba ejecutados en el ciclo.

Reporte de Obstáculos (RO)

Los obstáculos son riesgos del proyecto de prueba, que ocurrieron. Es una lista de los problemas que hacen la prueba menos eficiente. Para cada obstáculo se reporta el esfuerzo invertido en él. Dentro de los obstáculos a reportar se pueden encontrar [Kan01]:

- Cantidad de pruebas bloqueadas por defectos.
- Tentativas falladas de pasar las pruebas de humo.
- Cambios en las especificaciones durante la prueba.
- Pruebas que cambiaron por requerimientos modificados.
- Tiempo perdido en cosas del entorno (por ejemplo dificultades para obtener el equipo de prueba o configurar el sistema).

Resumen de Reunión (RR)

Minuta de las reuniones, donde se describen los participantes de la reunión, fecha en que se realizó la misma, lugar, temas tratados, acuerdos alcanzados, observaciones y lista de actividades a realizar a partir de la reunión y encargado de realizarlas. El resumen de la reunión debe ser aprobado por todos los participantes de la reunión.

Testware (TW)

El testware es el producto resultante de las actividades de prueba. Una posible descomposición de los componentes del Testware es la siguiente [Bla02]:

- **Herramientas**: Las herramientas necesarias para realizar las pruebas pueden comprender Sistemas Operativos, herramientas de automatización de las pruebas, manejadores de Bases de Datos, herramientas para la Gestión de las Pruebas y herramientas para la realización de reportes.
- **Inventario**: Reúne las funcionalidades a ser probadas.
- **Casos de Prueba**
- **Biblioteca de casos de prueba**: Para cada caso de prueba incluye: la configuración, las condiciones que deben cumplirse para poder ejecutar el caso de prueba y cómo volver al estado inicial.
- **Conjunto de Pruebas**: Organizan los casos de prueba. Incluyen la configuración del conjunto, los casos de prueba que los componen y cómo volver al estado inicial. Un mismo caso de prueba puede ejecutar en varias suites.
- **Resultados** (Logs): Los resultados pueden ser creados automáticamente por las herramientas o manualmente por los testers.
- **Reportes**: Se realizan a partir de los casos de prueba y los resultados de ejecutarlos.
- **Arquitectura de las pruebas**: Documento que muestra la estructura del Testware y las herramientas que aplican en el ambiente de pruebas. Muestra también la estructura del sistema a probar.

6.2 Roles

El Centro de ensayos de Software tiene definidos los roles y las responsabilidades de los involucrados en un proyecto de prueba. Para ProTest se tomó la misma definición de roles.

Los roles definidos son:
- Líder del Proyecto de Prueba
- Diseñador de Pruebas
- Tester
- Cliente
- Desarrollador o Contraparte Técnica.

El equipo de prueba está conformado por el Líder de Proyecto de Prueba, los Diseñadores de Pruebas y los Testers.

En cada proyecto de prueba, existe un cliente que es quien contrata el servicio de prueba. Un proyecto de prueba independiente puede ser contratado por:
- Quien va a comprar el producto de software
- Quien desarrolla el producto de software

A continuación se describen los requisitos, competencias y responsabilidades para cada rol definido. Los requisitos y competencias para el equipo de prueba han sido extractados del documento de Competencias del CES [CES06]. Para todos los roles se presentan las principales actividades en las que están involucrados y las responsabilidades en cada actividad.

Líder de Proyecto de Prueba

Los requisitos y competencias para el Líder del Proyecto de Prueba son:
- Capacidad de Liderazgo.
- Sólidos conocimientos en administración de proyectos.
- Excelente relación interpersonal.
- Fuerte experiencia de trabajo en equipo.
- Experto en técnicas de prueba.
- Excelente comprensión de las fases del proceso de prueba.
- Sólidos conocimientos en programación, tecnologías de bases de datos y sistemas operativos.
- Fuerte experiencia en Ingeniería de Requerimientos.

En la Tabla 15 se describen las responsabilidades del Líder de Prueba.

Actividad	Responsabilidades
I1 – Definición del Servicio	Definir el alcance, los criterios de aceptación para el proyecto de prueba y los criterios de finalización del mismo del proyecto de prueba junto con el Cliente y Desarrolladores
I2 – Revisión Preliminar de Requerimientos	Estudiar los principales requerimientos del producto junto con el Cliente y el Diseñador para realizar la propuesta de servicio
I3 – Análisis Preliminar de Riesgo del Producto	Identificar junto con el Cliente y los Desarrolladores las funcionalidades mas riesgosas y las que tengan mayor probabilidad de tener defectos

Actividad	Responsabilidades
P1 – Negociación con el Cliente	Negociar con el Cliente y los Desarrolladores el alcance definitivo y la agenda de los ciclos de prueba
P2 – Revisión de Requerimientos	Estudiar las especificaciones para ver si es posible generar las pruebas a partir de ellas
P3 – Análisis de Riesgo del Producto	Identificar junto con el Cliente y los Desarrolladores la prioridad de las funcionalidades incluidas en el alcance
P5 – Definición de los Ciclos de Prueba	Definir junto con el Cliente y los Desarrolladores la agenda de los ciclos de prueba y las funciones que serán probadas en cada ciclo.
P7 – Planificación de las Pruebas	Planificar el proyecto, definiendo la estrategia de Prueba y las técnicas a usar en el Proyecto Planificar el armado del ambiente de pruebas junto con el Cliente y los especialistas técnicos necesarios
P8 – Definición del Proceso de Incidentes	Definir junto con el Cliente y Desarrolladores la forma en que se reportan los incidentes encontrados y quienes serán los responsables por parte del cliente de validar y corregir
S1 – Planificación del Ciclo	Planificar en detalle el ciclo de prueba y las actividades a realizar por el equipo de pruebas
S3 – Seguimiento y Control del Ciclo	Líderar técnicamente el proyecto. Supervisar el equipo de prueba. Administrar costos y procesos. Implementar el proceso de prueba Seguir el progreso del proyecto de prueba y reportar el porcentaje de definición y ejecución de casos de prueba, así como la cantidad de incidentes detectados por caso de prueba y totales Interactuar con el Cliente en todas las etapas técnicas de la ejecución del proyecto Asegurar la calidad de todos los productos del proyecto de prueba Validar el armado del ambiente de prueba y su preservación
S4 – Evaluación de las Pruebas	Evaluar las pruebas del ciclo y definir cuando es necesario generar nuevos casos de prueba
V1 – Evaluación de la Satisfacción del cliente	Evaluar la satisfacción del Cliente con el proyecto de prueba y los resultados de las pruebas
V2 – Ajustes y Mejoras del Proceso de Prueba	Sugerir mejoras en el proceso de prueba.
V3 - Reporte Final del Proyecto de Prueba	Elaborar el informe final del proyecto de prueba
G1 – Estimación de Tareas	Estimar el esfuerzo que insumirá cada actividad y pedir al resto del equipo que estime sus tareas
G2 – Reporte de Esfuerzo	Reportar el esfuerzo que insume cada actividad y controlar que el resto del equipo reporte su esfuerzo

Tabla 15 – Responsabilidades del Líder de Prueba

Diseñador de Pruebas

Los requisitos y competencias para el Diseñador de Pruebas son:

- Experto en técnicas de prueba.
- Sólidos conocimientos en diseño de casos de prueba.
- Sólidos conocimientos en Ingeniería de Requerimientos.
- Excelente comprensión de las fases del proceso de prueba.
- Sólidos conocimientos en programación, tecnologías de bases de datos y sistemas operativos.
- Excelente experiencia en Pruebas de Software.
- Buen manejo de la expresión escrita.
- Excelente relación interpersonal.
- Experiencia de trabajo en equipo.

En la Tabla 16 se describen las responsabilidades del Diseñador de Pruebas.

Actividad	Responsabilidades
I2 – Revisión Preliminar de Requerimientos	Estudiar los principales requerimientos del producto junto con el Cliente y el Diseñador para realizar la propuesta de servicio
I3 – Análisis Preliminar de Riesgo del Producto	Identificar junto con el Cliente y los Desarrolladores las funcionalidades mas riesgosas y las que tengan mayor probabilidad de tener defectos
P2 – Revisión de Requerimientos	Definir la estrategia para generar los datos de prueba junto con la contraparte técnica del Cliente Verificar la calidad de los requerimientos para realizar las pruebas Detectar los problemas de especificación de requerimientos y elaborar propuestas para su mejora
P3 – Análisis de Riesgo del Producto	Definir junto con el Cliente las prioridades de las funcionalidades en cada ciclo de prueba
P4 – Explorar el producto	Estudiar el producto para conocerlo rapidamente
P6 – Definición del Testware	Definir los artefactos que serán generados durante el proyecto de prueba y cómo serán organizados
P7 – Planificación de las Pruebas	Ayudar al Líder de Proyecto a planificar el proyecto de prueba
P8 – Definición del Proceso de Incidentes	Definir junto con el Cliente y Desarrolladores la forma en que se reportan los incidentes encontrados y quienes serán los responsables por parte del cliente de validar y corregir
S1 – Planificación del Ciclo	Asistir al Líder de Proyecto en la planificación del ciclo.
S2 – Administración de la Configuración	Administrar los elementos definidos en el Testware durante todo el proyecto junto con el tester
S3 – Seguimiento y Control del Ciclo	Coordinar el equipo de testers y asistirlos en los momentos necesarios Coordinar reuniones técnicas con la Contraparte técnica del cliente o el equipo de desarrollo. Generar reportes de avance
S4 – Evaluación de las Pruebas	Analizar los resultados de las pruebas realizadas Elaborar los reportes de prueba Seguir la ejecución de las pruebas

Actividad	Responsabilidades
D1 – Diseño del los Casos de Prueba	Definir criterios para realizar las pruebas Diseñar los casos de prueba Generar y mantener la matriz de trazabilidad entre requerimientos y pruebas
D2 – Validación de los Casos de Prueba	Validar con el Cliente y/o los Desarrolladores que los casos de prueba sean de valor para el negocio
D3 – Asignación de los Casos de Prueba	Definir junto con el equipo de testers, quién ejecutará las pruebas definidas
E4 – Reporte de Incidentes	Revisar los incidentes reportados por los testers
V2 – Ajustes y Mejoras al Proceso de Testing	Sugerir mejoras en el proceso de prueba.
G1 – Estimación de Tareas	Estimar el esfuerzo que insumirá cada actividad
G2 – Reporte de Esfuerzo	R eportar el esfuerzo que insume cada actividad

Tabla 16– Responsabilidades del Diseñador de Pruebas

Tester

Los requisitos y competencias para el Tester son:

- Buen conocimiento en desarrollo de casos de prueba
- Buen conocimiento en probar software y en herramientas automáticas.
- Sólidos conocimientos en base de datos y en programación.
- Buenos conocimientos en técnicas de prueba.
- Comprensión de las fases del proceso de prueba

En la Tabla 17 se describen las responsabilidades del Tester.

Actividad	Responsabilidades
P4 – Explorar el Producto	Explorar el Producto para aprender sobre él
P6 – Definición del Testware	Definir junto con el Diseñador los artefactos que serán generados durante el proyecto de prueba
C1 – Instalación de Herramientas	Instalar las herramientas definidas en el Testware
C2 – Instalación y Configuración	Instalar y configurar la versión a probar del producto
S1 – Planificación del Ciclo	Planificar las pruebas en conjunto con los Diseñadores
S2 – Administración de la Configuración	Administrar los elementos definidos en el Testware durante todo el proyecto
S3 – Seguimiento y Control del Ciclo	Seguir la ejecución de las pruebas
S4 – Evaluación de las Pruebas	Analizar los resultados de las pruebas realizadas Elaborar los reportes de prueba
D3 – Asignación de los Casos de Prueba	Definir junto con el equipo de testers y diseñadores, quién ejecutará las pruebas definidas
E1 – Pruebas de Humo	Ejecutar las pruebas de humo Registrar los resultados reales de la ejecución de las pruebas
E2 – Ejecución de las Pruebas	Ejecutar las pruebas Registrar los resultados reales de la ejecución de las pruebas
E3 – Testing Exploratorio	Realizar testing exploratorio Registrar los resultados del testing exploratorio
E4 – Reporte de Incidentes	Detectar y registrar los incidentes en el sistema de seguimiento de incidentes

Actividad	Responsabilidades
E5 – Validación de los Incidentes	Asegurarse de que el Cliente valide los incidentes reportados
E6 – Verificación de las Correcciones	Validar que el incidente fue solucionado en la nueva versión del producto y que puede ser cerrado.
V2 – Ajustes y Mejoras al Proceso de Testing	Sugerir mejoras en el proceso de prueba.
V4 – Archivo del Testware	Dejar los elementos que componen el testware organizados de forma de poder ser accedidos fácilmente en otros proyectos
G1 – Estimación de Tareas	Estimar el esfuerzo que insumirá cada actividad
G2 – Reporte de Esfuerzo	Reportar el esfuerzo que insume cada actividad

Tabla 17– Responsabilidades del Tester

Cliente

El cliente es quien contrata el proyecto de prueba independiente, por lo que no tiene requisitos y competencias definidas. En la Tabla 18 se describen las responsabilidades del Cliente.

Actividad	**Responsabilidades**
I1 – Definición del Servicio	Definir el alcance, los criterios de aceptación para el proyecto de prueba y los criterios de finalización del mismo del proyecto de prueba junto con el Líder de Proyecto y Desarrolladores
I2 – Revisión Preliminar de Requerimientos	Brindar acceso a las fuentes de requerimientos existentes (documentos de requerimientos, manuales, documentación técnica, usuarios del sistema, etc.)
I3 – Análisis Preliminar de Riesgo del Producto	Identificar junto con el Líder de Proyecto las funcionalidades mas riesgosas y las que tengan mayor probabilidad de tener defectos
P1 – Negociación con el Cliente	Negociar con el Líder de Proyecto y los Desarrolladores el alcance definitivo y la agenda de los ciclos de prueba Definir los interlocutores del Cliente con el equipo de prueba
P2 – Revisión de Requerimientos	Brindar acceso a las fuentes de requerimientos existentes (documentos de requerimientos, manuales, documentación técnica, usuarios del sistema, etc.)
P3 – Análisis de Riesgo del Producto	Identificar junto con el Líder de Proyecto y los Desarrolladores la prioridad de cada funcionalidad que compone el alcance
P5 – Definición de los ciclos de prueba	Definir junto con el Líder de Proyecto y los Desarrolladores la agenda de los ciclos de prueba y las funciones que serán probadas en cada ciclo.
P7 – Planificación de las Pruebas	Planificar el proyecto, definiendo la estrategia de Prueba y las técnicas a usar en el Proyecto Planificar el armado del ambiente de pruebas junto con el Líder de Proyecto y los Desarrolladores necesarios
P8 – Definición del Proceso de Incidentes	Acordar con el Líder de Proyecto el procedimiento a seguir en caso de aparición de defectos que impidan continuar con las pruebas Identificar quiénes serán los responsables por parte del cliente de validar y corregir los incidentes encontrados en las pruebas
D2 – Validación de los Casos de	Validar que los casos de prueba sean de valor para el negocio

Prueba	
Actividad	Responsabilidades
E5 – Validación de los Incidentes	Validar los incidentes reportados en el sistema de seguimiento
V1 – Evaluación de la Satisfacción del Cliente	Realizar la encuesta de satisfacción
V3 - Reporte Final del Proyecto de Prueba	Validar el informe final del proyecto de prueba

Tabla 18– Responsabilidades del Cliente

Desarrollador o Contraparte técnica

Al igual que el cliente, el desarrollador es un rol externo al equipo de pruebas, por lo que no tiene requisitos y competencias. En la Tabla 19 se describen las responsabilidades del Desarrollador.

Actividad	Responsabilidades
I1 – Definición del Servicio	Definir el alcance, los criterios de aceptación para el proyecto de prueba y los criterios de finalización del mismo del proyecto de prueba junto con el Líder de Proyecto y el Cliente
I3 – Análisis Preliminar de Riesgo del Producto	Identificar junto con el Líder de Proyecto y el Cliente las funcionalidades mas riesgosas y las que tengan mayor probabilidad de tener defectos
P1 – Negociación con el Cliente	Negociar con el Líder de Proyecto y el Cliente el alcance definitivo y la agenda de los ciclos de prueba Definir los interlocutores con el equipo de prueba
P3 – Análisis de Riesgo del Producto	Identificar las áreas del producto que tienen más probabilidad de contener defectos
P5 – Definición de los ciclos de prueba	Definir los ciclos de prueba acordes con las fechas del plan de desarrollo
P8 – Definición del Proceso de Incidentes	Identificar quiénes serán los responsables por parte del cliente de validar y corregir los incidentes encontrados en las pruebas
D2 – Validación de los Casos de Prueba	Validar los casos de prueba y sus datos
E6 – Verificación de las Correcciones	Realizar el seguimiento y validación de los incidentes encontrados Responsabilizarse por la corrección de incidentes

Tabla 19– Responsabilidades del Desarrollador

Capítulo 7

CASO DE ESTUDIO

En este capítulo se presenta la aplicación de ProTest en este trabajo en un proyecto de prueba. Se describe cómo fueron seguidas las etapas del proceso de prueba funcional definido para la prueba de una aplicación de gestión.

En la sección 7.1 se describe brevemente el producto a probar. En la sección 7.2 se presenta la experiencia de utilizar ProTest en un proyecto de prueba real, describiendo cada una de las etapas del proceso: Estudio Preliminar, Planificación, Ciclos de Prueba y Evaluación. En la sección 7.3 se presentan las conclusiones y mejoras propuestas a ProTest luego de su aplicación.

7.1 Producto a probar

El producto de software usado en el caso de estudio es una aplicación de gestión que está en producción hace varios años, al cual se le realiza mantenimiento y mejoras. Existían dos versiones de la misma aplicación instaladas en clientes distintos, en el transcurso del proyecto de prueba, ambas versiones se unificaron en una sola aplicación. La documentación al comenzar el proyecto de prueba era un manual de usuario de la aplicación que no estaba actualizado.

El cliente es miembro de la empresa que desarrolla el producto. La empresa está interesada en mejorar su metodología de prueba, especialmente en formalizar su proceso de diseño y ejecución de las pruebas además de realizar una evaluación externa del producto.

Por motivos de confidencialidad, no se presenta el nombre de la empresa, ni de la aplicación, ni del cliente. Por la misma razón tampoco se muestran las funcionalidades a probar ni los casos de prueba generados. Sin embargo, el cliente está de acuerdo en mostrar en el presente caso de estudio los datos reales, sin necesidad de enmascararlos.

7.2 Aplicación de ProTest

A continuación se describe la aplicación de ProTest en el proyecto de prueba. Se organiza siguiendo las etapas del proceso: Estudio Preliminar, Planificación, Ciclo de Prueba y Evaluación.

7.2.1 Estudio Preliminar

La etapa de Estudio Preliminar tiene como objetivo estudiar la factibilidad de realizar el proyecto de prueba. A continuación se describen las actividades realizadas durante esta etapa.

La actividad **I1- Definición del Servicio** se llevó a cabo durante tres reuniones con el cliente en las cuales se explicó globalmente la aplicación y se describieron brevemente las principales funcionalidades. Se generaron los resúmenes de lo tratado en cada reunión, los cuales fueron enviados al cliente para su validación.

Para realizar la actividad **I2-Revisión Preliminar de Requerimientos**, a partir de la documentación existente del producto, se hizo una lista de las principales funcionalidades de la aplicación, basándose en los menús de la aplicación y las funcionalidades en cada menú, que aparecían en el manual de usuario de la aplicación. Se detectó que la documentación existente se encontraba incompleta, por lo que debían mejorarse las especificaciones, esto se tuvo en cuenta al momento de realizar el cronograma del proyecto de prueba. Las funcionalidades fueron listadas en el IP-Inventario de pruebas, se encontraron 42 funcionalidades durante esta actividad. El inventario de pruebas fué validado por el cliente.

En la actividad I2-Revisión Preliminar de Requerimientos se encontraron 42 funcionalidades, las cuales fueron priorizadas en la actividad I3- Análisis Preliminar de Riesgo del Producto. A partir de esta información y las necesidades del cliente se definió el alcance para las pruebas con 22 funcionalidades de las 42 encontradas. En el primer cronograma para las pruebas se definió que el proyecto de prueba duraría dos meses, ya que la agenda de desarrollo tenía previsto generar una versión por mes. Se define que las pruebas serían realizadas en el Laboratorio de Testing Funcional del Centro de Ensayos. Por el

equipo de prueba participaron: un líder de proyecto, un diseñador de casos de prueba y un tester. En la Tabla 20 se muestra la cantidad de funcionalidades definidas en el alcance del proyecto de prueba, discriminadas por prioridad, complejidad y criticidad.

Primer Alcance			
Prioridad	**Complejidad**	**Criticidad**	**# Func.**
Alta	Alta	Alta	1
Alta	Media	Alta	15
Alta	Baja	Alta	4
Alta	Baja	Media	1
Media	Media	Alta	1
		Total	**22**

Tabla 20- Alcance en el Estudio Preliminar

Para realizar la actividad **I3- Análisis Preliminar de Riesgo del Producto**, en las reuniones mantenidas con el cliente, se le explicó la importancia de realizar el análisis de riesgo del producto y que a partir de dicho análisis se definiría el alcance de las pruebas. A partir de la primera versión del IP-Inventario de Pruebas, el cliente modificó algunas funcionalidades de la lista y les asignó prioridad, criticidad e importancia. En este caso, el cliente era la empresa que desarrolló el producto, por lo que los responsables del cliente conocían qué partes del producto eran más complejas o tenían mayor probabilidad de tener defectos.

A partir de esta información, se realizó la PS- Propuesta de Servicio del proyecto de prueba. Esta propuesta incluyó el cronograma de las pruebas, el alcance y la cotización del proyecto de prueba. Las actividades necesarias para realizar la cotización no son parte de ProTest. El cliente dió el visto bueno a la propuesta, momento en el cual se comenzó la siguiente etapa: Planificación.

Esfuerzo de la Etapa: Estudio Preliminar

En la Tabla 15 se muestra el esfuerzo invertido en la etapa Estudio Preliminar por parte del equipo de prueba, en cada actividad realizada.

Estudio Preliminar			
Horas por Actividad	**Líder**	**Diseñador**	**Total**
I1-Definición del Servicio	6,5	2	8,5
I2-Revisión Preliminar de Requerimientos	2,5	11,5	14
I3-Análisis Preliminar de Riesgo	3	3	6
Total	**12**	**16,5**	**28,5**

Tabla 21– Esfuerzo durante el Estudio Preliminar

7.2.2 Planificación

Se tuvieron dos reuniones de negociación con el cliente durante la etapa de Planificación. En estas reuniones se realizaron a la vez las actividades:

- P1 - Negociación con el Cliente
- P3 - Análisis de riesgo
- P5 - Definición de los Ciclos de Prueba
- P8 - Definición del Proceso de Incidentes.

Como resultado de estas actividades, se definió que se tendrían reuniones semanales con el cliente y que se mantendría comunicación fluida por correo electrónico y mensajería electrónica instantánea. Se identificaron los responsables por parte del cliente de tomar las decisiones de alto nivel, de contestar dudas respecto al producto, de decidir sobre los cambios del alcance de las pruebas a lo largo del proyecto, de validar los casos de prueba y los resultados de su ejecución. Las versiones del producto serían instaladas por el equipo de prueba, se definió quién resolverá por parte del cliente los posibles problemas que pudieran ocurrir al momento de la instalación y configuración de la aplicación. Se definió entregar reportes semanales con los casos probados y los incidentes encontrados. Dado que la aplicación se encontraba instalada en el ambiente de producción, se contó con datos reales para las pruebas, se definió que era responsabilidad del cliente brindar un conjunto de datos reales y consistentes para realizar las pruebas. Se revisó el análisis de riesgo realizado durante la etapa Estudio Preliminar, donde no se encontraron cambios a lo definido en dicha etapa.

En la actividad **P2-Revisión de Requerimientos**, se revisó en detalle la documentación existente para entender el producto a probar. Dado que las especificaciones estaban incompletas, la revisión de requerimientos incluyó analizar y describir los requerimientos a probar. De esta forma, durante la etapa de planificación se mejoraron algunos de los requerimientos que serían probados durante el primer ciclo de prueba.

Dado que existía una versión ejecutable del producto a probar, se complementó esta actividad con la de **P4 -Exploración del Producto**. La exploración del producto ayudó al equipo de prueba a entender el dominio del negocio y a mejorar los requerimientos existentes, ahorrando tiempo del cliente en explicar los requerimientos para la aplicación.

Para la actividad **P5- Definición de los Ciclos de Prueba**, se definió junto con el cliente el comienzo y fin de cada ciclo y las funcionalidades que serían probadas en cada uno. En la Tabla 22 se describe la cantidad de funcionalidades planificadas para cada ciclo, discriminadas según Prioridad, Complejidad y Criticidad.

Alcance en Etapa de Planificación					
Prioridad	Complejidad	Criticidad	# Func.	# Ciclo 1	# Ciclo 2
Alta	Alta	Alta	1	0	1
Alta	Media	Alta	15	7	8
Alta	Baja	Alta	4	3	1
Alta	Baja	Media	1	1	0
Media	Media	Alta	1	0	1
		Total	**22**	**11**	**11**

Tabla 22- Alcance definido para la Etapa de Planificación

Para la actividad **P8 – Definición del Proceso de Incidentes**, se utilizó Bugzilla [Bug06] como herramienta para el reporte y seguimiento de los incidentes. En la Figura 48 se muestra el proceso de incidentes definido para el proyecto de prueba.

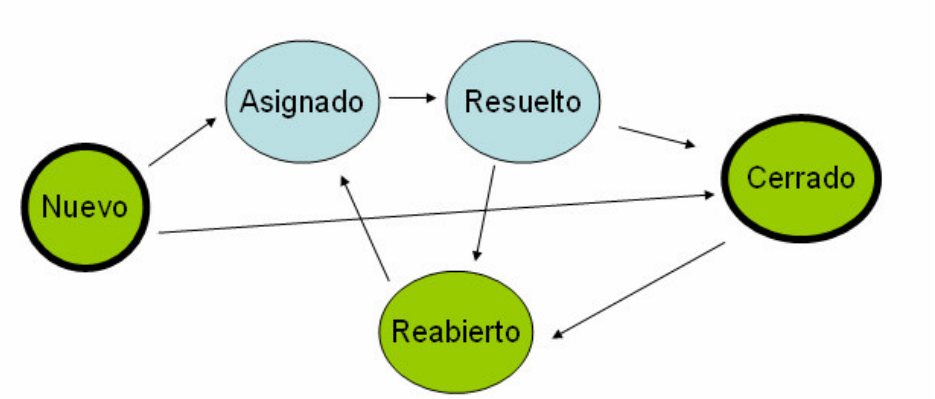

Figura 48– Estados por los que pasa un incidente

A continuación se describe brevemente cada estado:

- Nuevo: El incidente fue ingresado y debe ser procesado.
- Asignado: El incidente no esta aún resuelto, pero está asignado a una persona del equipo de desarrollo para su reparación.
- Reabierto: El incidente fue resuelto una vez, pero su resolución fue incorrecta.
- Resuelto: El incidente fue reparado por el equipo de desarrollo. Permanece en este estado hasta que se entrega la nueva versión del producto donde es reparado. Si las pruebas de regresión no encuentran el error pasa al estado Cerrado, sino al estado Reabierto.
- Cerrado: El incidente fue reparado y es cerrado.

El cliente debía pasar los incidentes al estado Asignado o Resuelto. Los estados Nuevo, Reabierto y Cerrado fueron usados por el equipo de prueba.

Se utilizó la siguiente clasificación para los incidentes encontrados durante las pruebas:

- Implementación incorrecta
- Mejora
- Interfaz de usuario
- Manejo de errores
- Validación faltante o incorrecta
- Funcionalidad no disponible
- Instalación o ambiente
- Salida anormal
- Seguridad
- Usabilidad
- Observación
- No clasificado

En la actividad **P7 – Planificación de las Pruebas** se realizó el plan de pruebas a partir de la información recolectada en las actividades P5 – Definición de los Ciclos de Prueba, P6 – Definición del Testware y P8 – Definición del Proceso de Incidentes. Se realizó la planificación de todo el proyecto de prueba la cual se muestra en la Tabla 23, la etapa de Planificación duraría 2 semanas, el Ciclo de Prueba 1 duraría 3

semanas, el Ciclo de Prueba 2 otras 3 semanas, la Evaluación del proyecto de prueba se realizaría en la última semana.

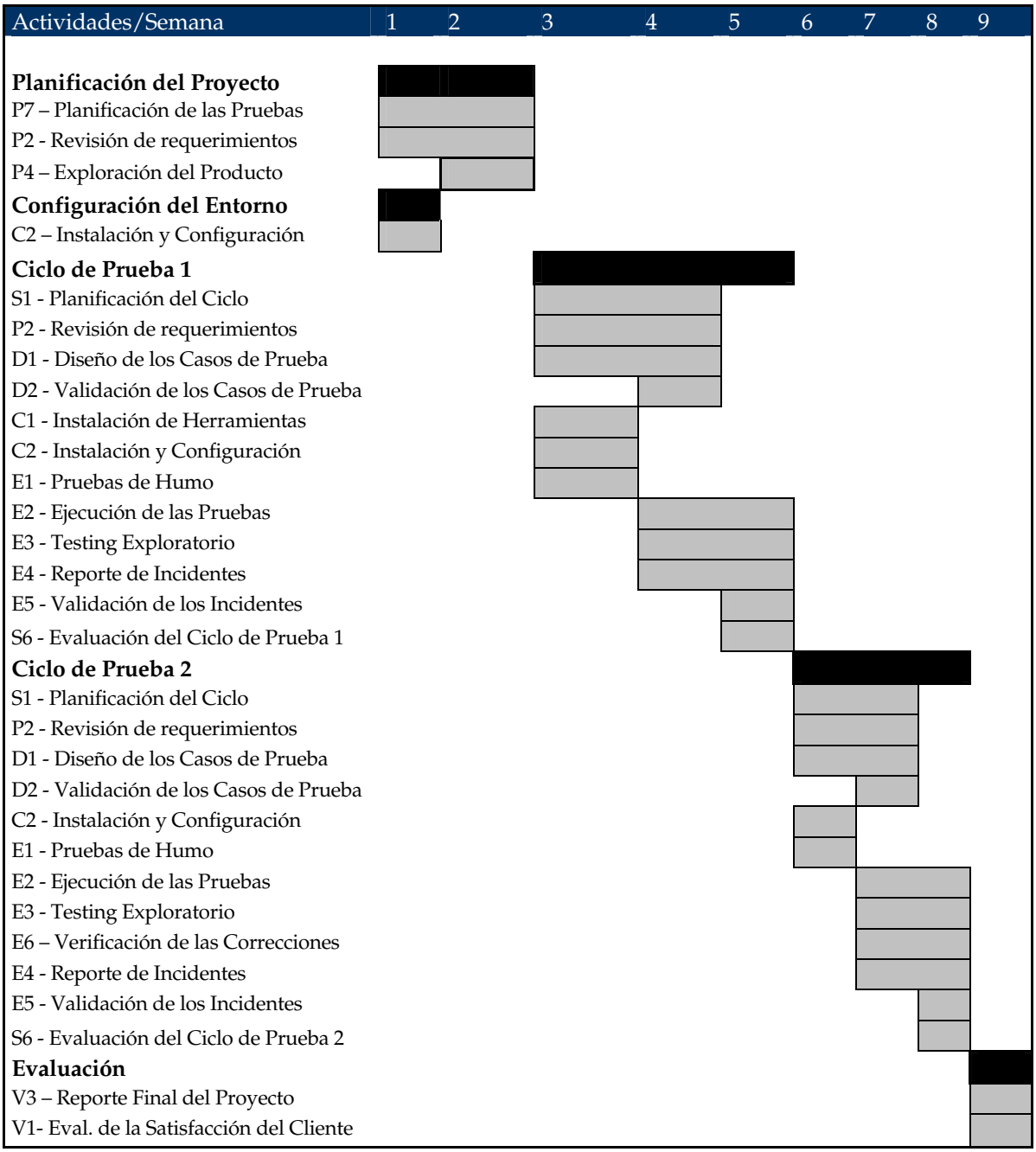

Tabla 23– Planificación del Proyecto de Prueba

El plan de pruebas fué realizado por el Líder de Proyecto y fué aprobado por el Cliente, antes de comenzar el primer ciclo de prueba. La definición del Plan de Pruebas resultó de gran utilidad para que

el cliente tenga visibilidad respecto al trabajo del proyecto de prueba independiente, así como lograr acuerdos respecto a responsabilidades y compartir un vocabulario común.

Esfuerzo de la Etapa: Planificación

En la Tabla 24 se muestra el esfuerzo invertido en la etapa de Planificación por parte del equipo de prueba, discriminado según las actividades realizadas en dicha etapa. En la actividad P1 – Negociación con el Cliente, se fusiona el esfuerzo de las actividades P3 - Análisis de riesgo, P5 – Definición de los Ciclos de Prueba y P8 – Definición del Proceso de Incidentes, ya que estas actividades fueron realizadas en forma simultánea en reuniones con el cliente. En la actividad P7-Planificación de las pruebas se incluye el esfuerzo de P6 – Definición del Testware.

La actividad C2 – Instalación y Configuración de la aplicación, corresponde a la sub-etapa Configuración del Entorno, esta actividad se realizó en la etapa de Planificación para poder explorar el producto.

Etapa: Planificación				
Horas por Actividad	**Líder**	**Diseñador**	**Tester**	**Total**
P1-Negociación con el Cliente	3	3	0	6
P2-Revisión de Requerimientos	9	14,7	0	23,7
P7- Planificación de las Pruebas	4	1	0	5
P4 -Exploración del Producto	0	4	0	4
C2- Instalación y Configuración	0	0	7,5	7,5
Total	**16**	**22,7**	**7,5**	**46,2**

Tabla 24 – Esfuerzo en la etapa de Planificación

7.2.3 Ciclos de Prueba

Se realizaron dos ciclos de prueba de la aplicación. En esta sección se describe las actividades realizadas en cada ciclo, organizadas según las sub-etapas que ocurren en cada ciclo.

7.2.3.1 Ciclo de Prueba 1

A continuación se describe como se llevaron a cabo las actividades de ProTest en el Ciclo de Prueba 1.

Seguimiento del Ciclo

El ciclo de prueba comenzó con una reunión con el cliente en la cual se realizó la actividad **P3 - Análisis de riesgo del producto**, donde se revisaron las prioridades y el alcance de las pruebas para el primer ciclo. Para realizar algunas de las pruebas, era necesario realizar determinados ciclos funcionales, que involucraban diseñar casos de prueba para funcionalidades asociadas. Se negoció el alcance para el ciclo de forma de tener esto en cuenta. En la actividad **S1- Planificación del ciclo**, se definió que el ciclo duraría una semana más de lo que estaba planificado en el Plan de Pruebas y que se agregarían estas nuevas funcionalidades. Se había planificado probar 11 funcionalidades en este ciclo, y se decidió probar 18 funcionalidades, de las cuales 10 habían sido planificadas desde un principio y las otras 8 se agregaron para cubrir los ciclos funcionales de prueba. Se acordó con el cliente que estas 8

funcionalidades agregadas no serían probadas con el mismo énfasis que las funcionalidades ya planificadas. También se acordó recortar el alcance del ciclo de prueba 2, por lo que se pasó de 11 funcionalidades a 8. En la Tabla 25 se muestran las funcionalidades planificadas para el ciclo 1 y las que realmente se probaron.

Alcance para Ciclo 1				
Prioridad	Complejidad	Criticidad	# Plan. Ciclo 1	# Real Ciclo1
Alta	Media	Alta	7	7
Alta	Baja	Alta	3	2
Alta	Baja	Media	1	1
Media	Media	Media	0	1
Media	Baja	Media	0	1
Baja	Baja	Media	0	6
		Total	**11**	**18**

Tabla 25 – Alcance planificado versus real para Ciclo 1

La actividad **S5 – Seguimiento y Control del Ciclo** se realizó mediante reuniones semanales del equipo de prueba. Durante todo el ciclo de prueba se realizaron las actividades: **G2 - Reporte de Esfuerzo** donde se reportó el esfuerzo realizado en el ciclo y **S3 - Administración de la Configuración** donde se puso bajo control de configuración todos los elementos generados por el proyecto de prueba definidos en el Testware.

Configuración del Entorno

Para el primer ciclo de pruebas se tenía planificado comenzar las pruebas con una nueva versión. Debido a que se atrasó la generación de la versión a ser probada, se definió junto con el cliente comenzar las pruebas con la versión que fue instalada para la exploración durante la etapa de Planificación, por lo que la actividad **C2 – Instalación y Configuración** no se realizó para el primer ciclo. En la actividad **C1- Instalación de Herramientas** se instalaron las herramientas necesarias para la gestión de las pruebas, se utilizó para la administración de los casos de prueba la herramienta DocTesting [Doc06] y Bugzilla [Bug06] para el reporte y seguimiento de incidentes.

Diseño de las pruebas

Dado que durante la etapa de Planificación se habían mejorado los requerimientos de algunas de las funcionalidades a probar en el primer ciclo, se comienza el Diseño de las pruebas con las actividades **D1- Diseño de las pruebas** y en paralelo se sigue realizando **P2 –Revisión de requerimientos** del resto de las funcionalidades del ciclo.

Los casos de prueba se generaron utilizando las técnicas de caja negra de partición de equivalencia y valor límite. Se creó una matriz de cubrimiento para mostrar la correspondencia entre las funcionalidades y los casos de prueba generados. Los casos de prueba fueron validados en una reunión con el cliente. En la Tabla 26 se muestran para cada funcionalidad incluida en el alcance del ciclo de prueba 1, cuántos casos de prueba se generaron y cuántos incidentes se encontraron al ejecutar las pruebas con esos casos de prueba.

Casos Definidos por Funcionalidad – Ciclo de Prueba 1					
Prioridad	**Complejidad**	**Criticidad**	**Funcionalidad**	**# Casos**	**# Incidentes**
Alta	Baja	Media	Funcionalidad 1	4	0
Alta	Media	Alta	Funcionalidad 2	3	1
Alta	Media	Alta	Funcionalidad 3	9	3
Alta	Baja	Alta	Funcionalidad 4	3	0
Alta	Media	Alta	Funcionalidad 5	9	4
Alta	Media	Alta	Funcionalidad 6	4	1
Alta	Media	Alta	Funcionalidad 7	13	4
Alta	Media	Alta	Funcionalidad 8	10	5
Alta	Baja	Alta	Funcionalidad 9	26	11
Alta	Media	Alta	Funcionalidad 10	1	1
Baja	Baja	Media	Funcionalidad 11	4	1
Media	Media	Media	Funcionalidad 12	1	2
Baja	Baja	Media	Funcionalidad 13	1	0
Media	Baja	Media	Funcionalidad 14	3	0
Baja	Baja	Media	Funcionalidad 15	9	0
Baja	Baja	Media	Funcionalidad 16	2	1
Baja	Baja	Media	Funcionalidad 17	2	0
Baja	Baja	Media	Funcionalidad 18	2	4
			Total	**106**	**38**

Tabla 26 – Casos de Prueba e Incidentes encontrados por funcionalidad

Ejecución de las Pruebas

Dado que la versión probada en el ciclo 1 es la misma que se instaló en la etapa de Planificación, no fue necesario realizar la actividad **E1-Pruebas de Humo**, pues ya se sabía que las funcionalidades básicas funcionaban en el producto. Se ejecutaron los casos de prueba en la actividad **E2 – Ejecución de las Pruebas** y se realizó la actividad **E3 – Testing Exploratorio**. Los incidentes encontrados se reportaron en la herramienta de seguimiento de incidentes Bugzilla y luego fueron validados por el cliente. En la Tabla 27 se muestran los incidentes encontrados, discriminados por tipo de incidente.

Incidentes por Tipo	Ciclo 1
Funcional - Implementación Incorrecta	10
Interfaz de Usuario	7
Mensajes erróneos o falta de mensaje	20
Validación faltante o incorrecta	1
Total	38

Tabla 27 – Incidentes por tipo para el Ciclo 1

En la Tabla 28 se muestran los incidentes según la criticidad asignada a los mismos por el Cliente.

Incidentes según criticidad	Ciclo 1
Alta	3
Media	2
Baja	33
Total	38

Tabla 28 – Incidentes según criticidad para el Ciclo 1

Al terminar la sub-etapa Ejecución de las Pruebas, se realizó la actividad **S6 – Evaluación de las Pruebas**, donde se evalúan las actividades del Ciclo de Prueba. En la Tabla 29 se muestra el cronograma real seguido en el ciclo de prueba 1, las principales diferencias con lo planificado son:

- La actividad S1- Planificación del Ciclo se realizó en una semana y no en dos, como estaba planificado.
- Las actividades C2 – Instalación y configuración y E1- Pruebas de Humo no fueron realizada s debido a que se probó la versión instalada en la etapa de Planificación.
- Las actividades P2 – Revisión de Requerimientos, D1-Diseño de las pruebas y D2 – Validación de los Casos de Prueba llevaron dos semanas más de lo planificado en el Plan de Pruebas. La actividad S1-Planificación del Ciclo llevó una semana más de lo planificado.
- Las actividades E2 –Ejecución de las Pruebas, E3-Testing Exploratorio y E4-Reporte de Incidentes llevaron una semana más de lo planificado.

Se realizó el informe de Avance del Ciclo, resumiendo el resultado de la ejecución de las pruebas. Dicho resultado se discutió con el cliente en una reunión donde se presentó el Informe de Avance del Ciclo y se decidió dar por terminado el Ciclo 1. En la Tabla 30 se muestra el resultado de la ejecución del Ciclo de Prueba 1.

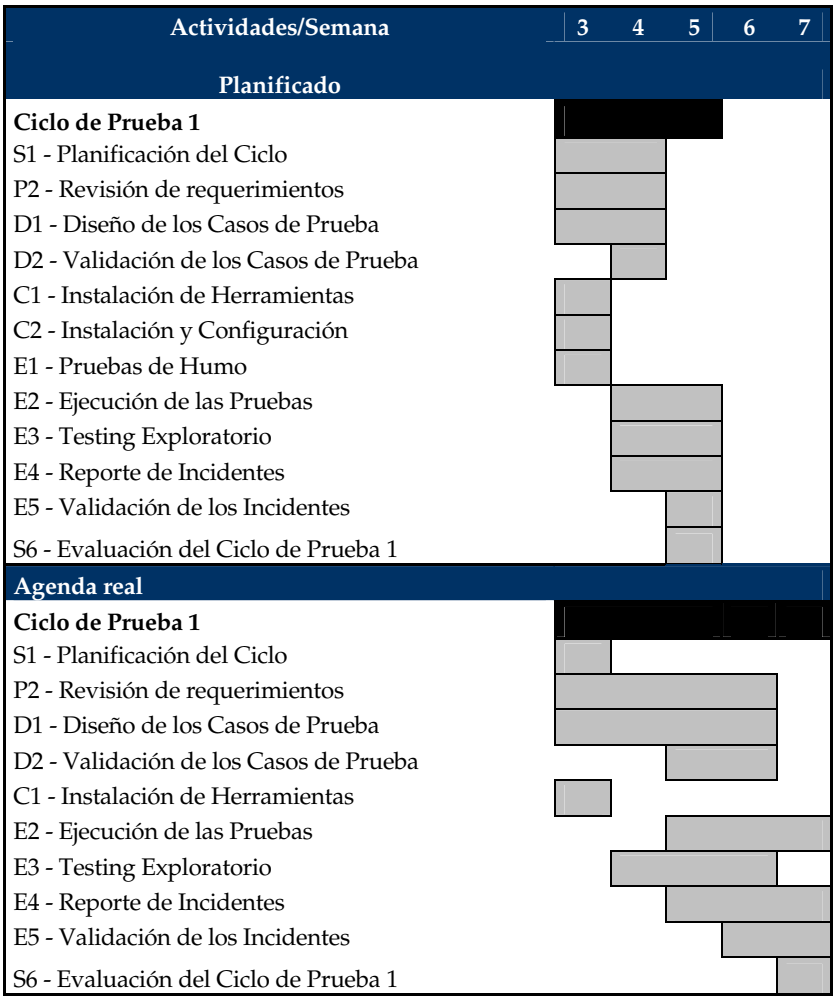

Tabla 29– Cronograma planificado versus real del Ciclo 1

Resumen Ejecución Ciclo de Prueba 1	
Funcionalidades	**18**
Casos de Prueba	**106**
Casos por Funcionalidad	**5,9**
Incidentes	**38**
Incidentes por Funcionalidad	**2,1**
Incidentes por caso ejecutado	**0,36**

Tabla 30 – Resumen de la ejecución del Ciclo de Prueba 1

Esfuerzo de la Etapa: Ciclo de Prueba 1

En la Tabla 31 se muestra el esfuerzo invertido en la etapa Ciclo de Prueba 1 por parte del equipo de prueba, discriminado por las actividades realizadas en dicha etapa.

Etapa: Ciclo de Prueba 1				
Horas por Actividad	Líder	Diseñador	Tester	Total
S1- Planificación del Ciclo	2	2		4
P3 - Análisis de Riesgo del Producto	2	2		4
S3 -Administración de la Configuración	0	3		3
S5- Seguimiento y Control del Ciclo	3	3		6
S6 - Evaluación del Ciclo de Prueba	16	3,5		19,5
C1 –Instalación de Herramientas			2	2
P2 – Revisión de Requerimientos		9,5		9,5
D1 – Diseño de los Casos de Prueba		49,5		49,5
D2 – Validación de los Casos de Prueba	2	2		4
E2 – Ejecución de las Pruebas			41,5	41,5
E3 – Testing Exploratorio			8	8
E4 – Reporte de Incidentes			8	8
E5 – Validación de los Incidentes		2	2	4
Total	**25**	**76,5**	**61,5**	**163**

Tabla 31 – Esfuerzo para la etapa Ciclo de Prueba 1

7.2.3.2 Ciclo de Prueba 2

A continuación se describe como se llevaron a cabo las actividades de ProTest en el Ciclo de Prueba 2.

Seguimiento del Ciclo

Al terminar el Ciclo de Prueba 1, el cliente no estaba en condiciones de liberar la próxima versión de la aplicación, ya que el desarrollo de la misma se encontraba atrasado. La nueva fecha para la entrega de la nueva versión sería en quince días más, por lo que se decidió atrasar el comienzo del ciclo en dos semanas. Dos semanas después el desarrollo continuaba atrasado, por lo que se decidió comenzar el ciclo de prueba de todos modos, se realizaría el diseño de los casos de prueba del ciclo y se esperaría a que estuviera la nueva versión disponible. La versión estuvo disponible tres semanas después, momento en el cual se comenzó la ejecución de las pruebas.

En el ciclo de prueba 1 se había definido recortar el alcance del segundo ciclo debido a que se agregaron funcionalidades en el ciclo 1. En la Tabla 32 se muestra el alcance planificado para el ciclo 2 y el alcance real, respecto a las nuevas funcionalidades a probar en este ciclo.

La nueva versión a probar unifica dos versiones existentes de la aplicación, por lo que el cliente estaba muy interesado en volver a ejecutar algunas pruebas debido a que no estaba seguro del funcionamiento de algunas funcionalidades luego de la unificación. Las pruebas de regresión no fueron planificadas porque no se sabía cuáles serían los incidentes encontrados por el equipo de prueba ni cuáles de ellos serían realmente corregidos para esta nueva versión. Se definió que se volverían a ejecutar 42 casos de pruebas de regresión. En la Tabla 33 pueden verse los casos de prueba de regresión ejecutados y los incidentes encontrados.

Alcance de nuevas funcionalidades para el Ciclo 2				
Prioridad	**Complejidad**	**Criticidad**	**# Plan. Ciclo 2**	**# Real Ciclo 2**
Alta	Alta	Alta	1	1
Alta	Media	Alta	8	5
Alta	Baja	Alta	1	0
Media	Media	Alta	1	2
		Total	**11**	**8**

Tabla 32 - Alcance planificado versus real para el Ciclo 2

Configuración del Entorno

Se instaló la nueva versión de la aplicación sin problemas en el Laboratorio de Testing del CES, junto con la nueva versión el cliente entregó un documento con los cambios realizados y las funcionalidades incluidas en la nueva versión.

Diseño de las pruebas

Los requerimientos para las nuevas funcionalidades a probar en el Ciclo 2 fueron analizados y documentados y se diseñaron las pruebas en forma previa a tener la versión ejecutable del producto.

Para generar los casos de prueba se utilizaron las técnicas de caja negra de partición en clases de equivalencia y valor límite. Se creó una matriz de cubrimiento para mostrar la correspondencia entre la funcionalidad y los casos de prueba generados. Los casos de prueba fueron validados en una reunión con el cliente y por correo electrónico. En la Tabla 33 se muestran para cada funcionalidad incluida en el alcance del ciclo de prueba 2, cuántos casos de prueba se generaron y cuántos incidentes se encontraron al ejecutar las pruebas con esos casos de prueba.

Casos Definidos por Funcionalidad - Ciclo 2					
Complejidad	**Criticidad**	**Prioridad**	**Funcionalidad**	**Casos definidos**	**# Incidentes**
Alta	Alta	Media	Funcionalidad 19	6	0
Alta	Alta	Media	Funcionalidad 20	3	0
Alta	Media	Media	Funcionalidad 21	5	1
Alta	Alta	Media	Funcionalidad 22	4	1
Alta	Alta	Media	Funcionalidad 23	25	17
Alta	Alta	Media	Funcionalidad 24	9	5
Alta	Media	Media	Funcionalidad 25	3	0
Alta	Alta	Alta	Funcionalidad 26	2	1
			Total	**57**	**25**

Tabla 33 - Caso de Prueba e Incidentes encontrados por funcionalidad

Ejecución de las Pruebas

Se realizó la actividad **E1-Pruebas de Humo**. Dado que la aplicación contenía las funcionalidades básicas, se ejecutaron los casos de prueba definidos en la actividad **E2 – Ejecución de las Pruebas** y se realizó la actividad **E3 – Testing Exploratorio**. También se ejecutaron las pruebas de regresión definidas al comienzo del ciclo, en la Tabla 34 se muestra el resultado de su ejecución. En la Tabla 35 se muestran los incidentes encontrados, discriminados por tipo de incidente y en la Tabla 36 se muestran los incidentes según la criticidad asignada a los mismos por el cliente.

Casos de Regresión - Ciclo 2					
Complejidad	**Criticidad**	**Prioridad**	**Funcionalidad**	**Casos ejecutados**	**# Incidentes**
Alta	Media	Alta	Funcionalidad 2	2	0
Alta	Media	Alta	Funcionalidad 3	3	1
Alta	Media	Alta	Funcionalidad 5	3	1
Alta	Media	Alta	Funcionalidad 7	4	2
Alta	Media	Alta	Funcionalidad 8	5	3
Alta	Baja	Alta	Funcionalidad 9	12	4
Baja	Baja	Media	Funcionalidad 11	4	0
Baja	Baja	Media	Funcionalidad 15	9	5
			Total	**42**	**16**

Tabla 34 – Casos de Prueba de Regresión ejecutados e Incidentes encontrados

Incidentes - Ciclo de Prueba 2		
Incidentes por Tipo	**Ciclo 2**	**Regresión**
Funcional - Implementación Incorrecta	7	3
Interfaz de Usuario	12	13
Mensajes erróneos y Falta de Mensaje.	1	0
Validación faltante o incorrecta	5	0
Total	**25**	**16**

Tabla 35 – Incidentes por Tipo para el Ciclo 2

Incidentes según criticidad	**Ciclo 2**	**Regresión**
Alta	3	2
Media	4	0
Baja	18	14
Total	**25**	**16**

Tabla 36 – Incidentes según criticidad para el Ciclo 2

Al terminar la Ejecución de las Pruebas, se realiza la actividad S6 – Evaluación de las Pruebas, donde se evalúan las actividades del Ciclo de Prueba. En la Tabla 37 se muestra el cronograma real seguido en el ciclo de prueba 2, las principales diferencias con lo planificado son:

- El ciclo de prueba 1 duró dos semanas más de lo planificado, por lo que el ciclo de prueba 2 debería haber comenzado en la semana 8 y no en la semana 10.

- La actividad S1- Planificación del Ciclo se realizó en una semana y no en dos, como estaba planificado.
- Las actividades P2 – Revisión de Requerimientos y D1-Diseño de los casos de prueba comenzaron cuatro semanas más tarde de lo planificado en el Plan de Pruebas y llevaron una semana más de lo planificado. Esto se debe a que la versión de la aplicación no estaba lista y se continuaron generando casos de prueba en esas semanas.
- Las actividades que tienen que ver con la ejecución de las pruebas (E2,E3,E4 y E6), se hicieron en dos semanas como estaba planificado a pesar del atraso debido a la falta de la versión.
- La actividad E5 - Validación de los Incidentes, que se había planificado para una semana, se realizó en las dos semanas de la ejecución de las pruebas, debido a que el sistema web de seguimientos de incidentes de Bugzilla permitía al cliente ver diariamente los incidentes y validarlos.

Se realizó el informe de Avance del Ciclo, resumiendo el resultado de la ejecución de las pruebas. En la Tabla 38 se muestra el resultado de la ejecución del Ciclo de Prueba 2, incluye las pruebas de regresión ejecutadas para el ciclo.

Actividades/Semana	6	7	8	9	10	11	12	13	14
Planificado									
Ciclo de Prueba 2									
S1 - Planificación del Ciclo									
P2 – Revisión de requerimientos									
D1 - Diseño de los Casos de Prueba									
D2 - Validación de los Casos de Prueba									
C2 - Instalación y Configuración									
E1 - Pruebas de Humo									
E2 - Ejecución de las Pruebas									
E3 - Testing Exploratorio									
E6 – Verificación de las Correcciones									
E4 - Reporte de Incidentes									
E5 - Validación de los Incidentes									
S6 - Evaluación del Ciclo de Prueba 2									
Agenda real									
Ciclo de Prueba 2									
S1 - Planificación del Ciclo									
P2 – Revisión de requerimientos									
D1 - Diseño de los Casos de Prueba									
D2 - Validación de los Casos de Prueba									
C2 - Instalación y Configuración									
E1 - Pruebas de Humo									
E2 - Ejecución de las Pruebas									
E3 - Testing Exploratorio									
E6 – Verificación de las Correcciones									
E4 - Reporte de Incidentes									
E5 - Validación de los Incidentes									
S6 - Evaluación del Ciclo de Prueba 2									

Tabla 37 - Cronograma planificado versus real para el Ciclo 2

Resumen Ejecución Ciclo de Prueba 2	
Cantidad de Funcionalidades	**16**
Cantidad de Casos de Prueba	**99**
Casos por Funcionalidad	**6,2**
Cantidad de Incidentes	**41**
Incidentes por Funcionalidad	**2,6**
Incidentes por caso ejecutado	**0,4**

Tabla 38 – Resumen de la Ejecución del Ciclo 2

Esfuerzo de la Etapa: Ciclo de Prueba 2

En la Tabla 39 se muestra el esfuerzo invertido en la etapa Ciclo de Prueba 2 por parte del equipo de prueba, discriminado por las actividades realizadas en dicha etapa.

Etapa: Ciclo de Prueba 2				
Horas por Actividad	**Líder**	**Diseñador**	**Tester**	**Total**
S1- Planificación del Ciclo	2	2		**4**
P3 - Análisis de Riesgo del Producto	2	2		**4**
S5- Seguimiento y Control del Ciclo	3	3		**6**
S6 - Evaluación del Ciclo de Prueba	18	8		**26**
C2 – Instalación y Configuración			7	**7**
P2 – Revisión de Requerimientos	8,5	18,5		**27**
D1 – Diseño de los Casos de Prueba	8	29,5		**37,5**
D2 – Validación de los Casos de Prueba	2	2		**4**
E1 – Pruebas de Humo			4	**4**
E2 – Ejecución de las Pruebas			27	**27**
E3 – Testing Exploratorio			6	**6**
E4 – Reporte de Incidentes			8	**8**
E6 – Verificación de las Correcciones			13	**13**
Total	**43,5**	**65**	**65**	**173,5**

Tabla 39 – Esfuerzo en el Ciclo de Prueba 2

7.2.4 Evaluación del Proyecto

La etapa de Evaluación del Proyecto, comenzó mientras se estaba evaluando el Ciclo de Prueba 2, con la actividad **V3 – Reporte Final del Proyecto**, en el que se resume el proyecto de prueba.

En la Tabla 40 se muestra la planificación realizada en el Plan de Pruebas versus la agenda real del proyecto de prueba. El informe final del proyecto de prueba se le entregó al cliente en una reunión junto con el informe de avance del ciclo 2. El cliente estuvo de acuerdo con los informes presentados.

En esa misma reunión se realizó la actividad **V1 - Evaluación de la satisfacción del Cliente** y también se le pidió luego al cliente que completara una encuesta de satisfacción. En la Tabla 41 se muestran algunas de las preguntas realizadas y la respuesta del cliente. Se utiliza una escala del 1 al 5, donde 1 es Malo y 5 Excelente.

Etapas/Semana	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Plan del Proyecto															
Estudio Preliminar															
Planificación del Proyecto															
Ciclo de Prueba 1															
Ciclo de Prueba 2															
Evaluación															
Agenda Real del Proyecto															
Estudio Preliminar															
Planificación del Proyecto															
Ciclo de Prueba 1															
Ciclo de Prueba 2															
Evaluación															

Tabla 40 – Planificado versus real para el Proyecto de Prueba

Encuesta de Satisfacción al Cliente	
Pregunta	**Valor**
¿Cómo evalúa el análisis del producto y la priorización de funcionalidades realizada?	☐1 ☐2 ☒3 ☐4 ☐5
¿Considera adecuado el número de fases/ciclos definidos?	☐1 ☐2 ☐3 ☒4 ☐5
¿Está de acuerdo con los productos intermedios obtenidos?	☐1 ☐2 ☐3 ☒4 ☐5
¿Cuán efectivas le resultaron las reuniones realizadas al fin de cada fase/ciclo?	☐1 ☐2 ☐3 ☒4 ☐5
¿Cómo evalúa el nivel de información y participación que tuvo durante el proyecto?	☐1 ☐2 ☐3 ☒4 ☐5
¿Está de acuerdo con la frecuencia de las reuniones realizadas?	☐1 ☐2 ☐3 ☒4 ☐5
Evalúe la relevancia de los incidentes encontrados :	☐1 ☐2 ☒3 ☐4 ☐5
¿Cuántos incidentes encontrados eran ya de su conocimiento?	☐1 ☐2 ☐3 ☒4 ☐5
¿En qué medida los resultados del proyecto de prueba coinciden con su percepción de la calidad del producto?	☐1 ☐2 ☐3 ☒4 ☐5
¿En qué medida el trabajo realizado contribuyó al desarrollo de la metodología interna de su empresa?	☐1 ☐2 ☐3 ☒4 ☐5
¿En qué medida el trabajo realizado contribuyó a la mejora de la calidad de su producto?	☐1 ☐2 ☐3 ☒4 ☐5

Tabla 41 – Encuesta de Satisfacción del Cliente

Las gráficas que se muestran a continuación resumen el proyecto de prueba. En la Figura 49 se muestra la cantidad de funcionalidades probadas, los casos de prueba generados y los incidentes encontrados para cada ciclo de prueba, incluyendo las pruebas de regresión y el total para todo el proyecto.

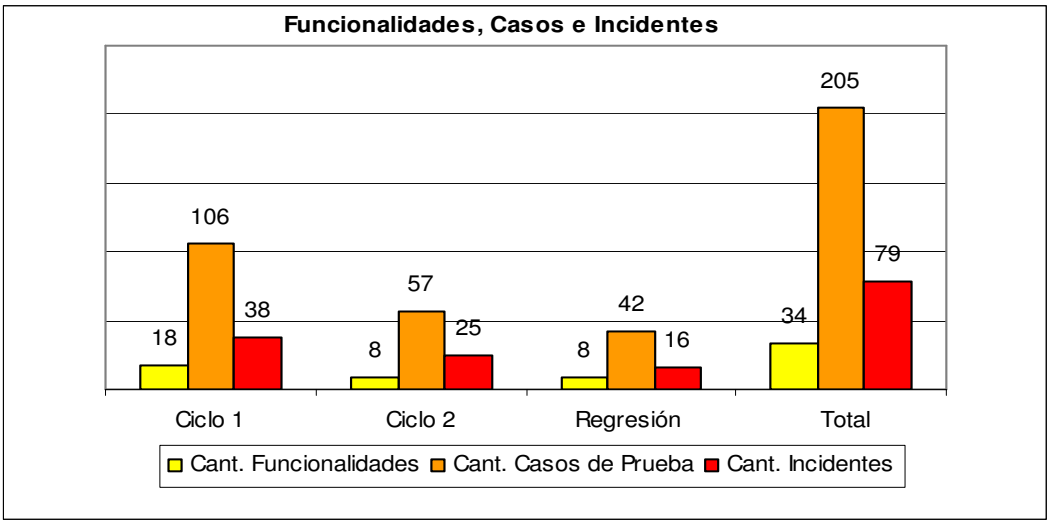

Figura 49 - Cantidad de Funcionalidades, Casos de Prueba e Incidentes del Proyecto

En la Figura 50 se muestra la cantidad de incidentes por Tipo de incidente definido para este proyecto de prueba: Implementación incorrecta, Interfaz de usuario, mensajes erróneos y validación faltante o incorrecta.

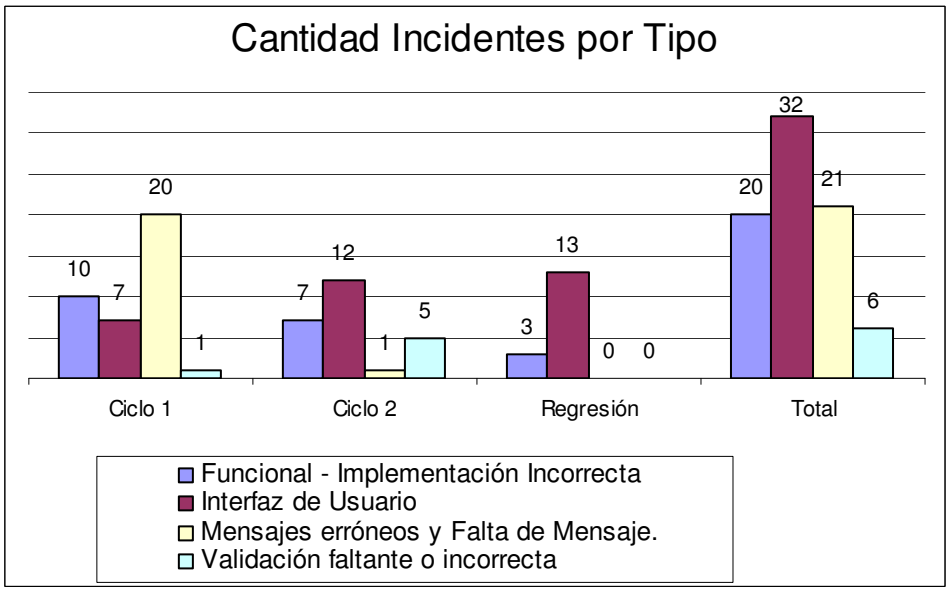

Figura 50 - Cantidad de Incidentes por Tipo

En la Figura 51 se muestran los incidentes encontrados en el proyecto de prueba, discriminados según la criticidad asignada por el cliente a cada uno. Se puede observar que la gran mayoría de los incidentes fueron de criticidad baja. De la evaluación realizada junto al cliente, surge que al estar la aplicación desde hace mucho tiempo en producción los principales incidentes ya han sido detectados y corregidos. Se puede apreciar en esta gráfica que los incidentes encontrados en la segunda versión del producto, esto

es los incidentes del ciclo 2 más los incidentes de regresión, son en total 42, mientras que en el ciclo 1 se encontraron 38 incidentes. Esto se debe a que la versión nueva de la aplicación incluía la fusión de dos versiones distintas existentes y este cambio introdujo nuevos incidentes en la aplicación.

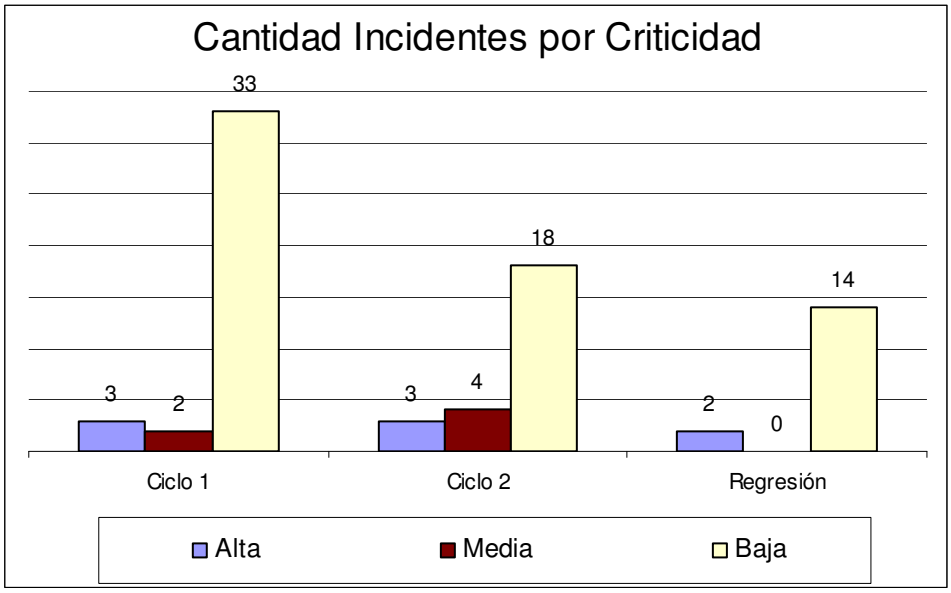

Figura 51 – Cantidad de Incidentes según Criticidad (Alta, Media, Baja)

En la Figura 52 se muestra el porcentaje de los incidentes según criticidad para cada versión y para las pruebas de regresión del ciclo 2. Se puede observar que en el segundo ciclo se incrementó el porcentaje de incidentes de prioridad alta encontrados, esto se debe a que, como se explicaba anteriormente, la aplicación sufrió un cambio bastante drástico en esta nueva versión.

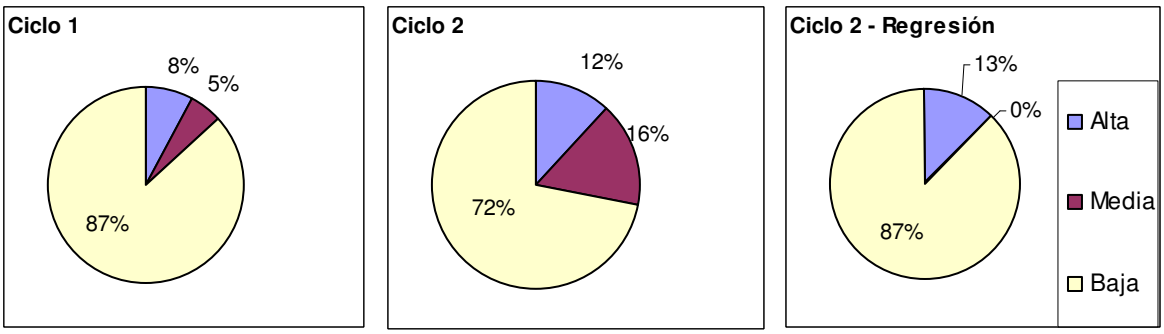

Figura 52 – Porcentaje de Incidentes según criticidad (Alta, Media, Baja)

Por último, en la Figura 53 se muestran para cada ciclo de prueba, los siguientes indicadores:

- Casos de Prueba por Funcionalidad: Se calcula como Total de Casos de Prueba / Total de Funcionalidades. Muestra la cantidad de casos de prueba generados en promedio por funcionalidad.
- Incidentes por funcionalidad: Se calcula como el Total de Incidentes / Total de Funcionalidades. Muestra el promedio de incidentes que tiene cada funcionalidad

- Incidentes por Caso de Prueba: Se calcula como el Total de Incidentes / Total de Casos de Prueba. Muestra el promedio de incidentes que encontró cada caso de prueba ejecutado.

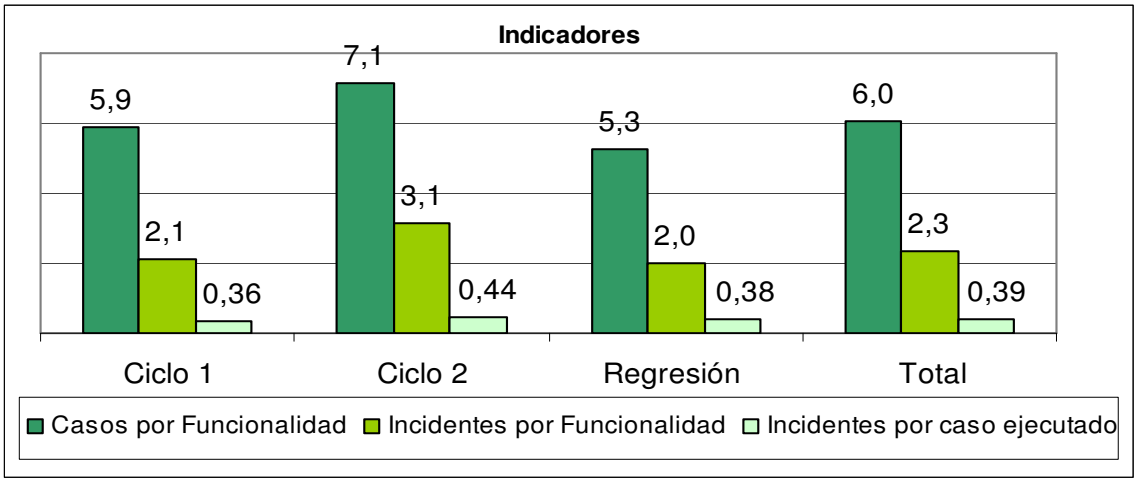

Figura 53 – Indicadores del Proyecto de Prueba

Esfuerzo de la Etapa: Evaluación del Proyecto

En la Tabla 42 se muestra el esfuerzo invertido en la etapa de Evaluación por parte del equipo de prueba, discriminado por las actividades realizadas en dicha etapa. La actividad V1-Evaluación de la satisfacción del cliente, contabiliza las horas de la reunión donde se mostraron los informes finales para el proyecto y se evaluó el proyecto junto con el cliente.

Etapa: Evaluación			
Horas por Actividad	**Líder**	**Diseñador**	**Total**
V1- Evaluación de la Satisfacción del Cliente	1,5	1,5	**3**
V3 – Reporte Final del Proyecto	10	7	**17**
Total	**11,5**	**8,5**	**20**

Tabla 42 – Esfuerzo en la etapa de Evaluación

7.3 Conclusiones y Ajustes

La puesta en práctica de ProTest en un proyecto de prueba real resultó un aporte significativo como guía para la realización de las pruebas del producto. Las características de ProTest permitieron realizar un proyecto de prueba que resultó de valor para quien contrató el servicio. En particular, cabe destacar las siguientes características de ProTest que resultaron valiosas durante el proyecto de prueba:

- Independencia del proceso de desarrollo: En ningún momento fué necesario conocer la forma de trabajo de la empresa de desarrollo ni tampoco ésta tuvo que cambiar su forma de trabajo para poder realizar pruebas de su producto con el equipo del CES. Los puntos de interacción entre el equipo de desarrollo y el equipo de prueba definidos en el proceso fueron suficientes para realizar el proyecto.
- Organización del proyecto de prueba en ciclos: Esto resultó una forma de planificación clara tanto para el equipo de pruebas como para el cliente. Desde el punto de vista del cliente, los ciclos de prueba y las funcionalidades a probar en cada ciclo definidos en la etapa de Estudio Preliminar permitieron realizar el seguimiento y control del proyecto de prueba. Al equipo de prueba el tener definidas las funcionalidades a probar en cada ciclo, le ayudó al momento de negociar cambios en el alcance y para decidir cuando terminar las pruebas.
- Análisis de riesgo del producto: Esto permitió organizar el proyecto en base a las prioridades del cliente, ayudando a la comunicación entre el cliente y el equipo de pruebas independiente
- Especificaciones de requerimientos incompletas: El proceso puede ser utilizado aún si no se cuenta con las especificaciones de los requerimientos antes de comenzar el proyecto de prueba. De este caso de estudio se desprende que los requerimientos pueden ser analizados y documentados a medida que se requiere probar las funcionalidades en los ciclos, teniendo en cuenta las prioridades asignadas en el análisis de riesgo del producto. El contar con una versión ejecutable del producto desde el principio ayudó a entender los requerimientos y describirlos sin la necesidad de tener demasiadas reuniones con el cliente.
- Testing exploratorio: Éste resultó una forma poco costosa de probar las funcionalidades. En esta experiencia no se encontraron incidentes importantes que no fueran también encontrados durante la ejecución de los casos de prueba planificados.

La colaboración del cliente para contestar dudas, priorizar funcionalidades, validar casos de prueba e incidentes fué esencial para poder realizar el proceso en forma exitosa. A continuación se describen algunos aspectos del proceso que resultaron positivos en la comunicación con el cliente:

- Plan de pruebas: Éste resultó de gran utilidad para que el cliente tenga visibilidad respecto al trabajo del proyecto de prueba independiente, así como para lograr acuerdos respecto a responsabilidades y compartir un vocabulario común.
- Informes de avance: La realización de los mismos al finalizar cada ciclo de prueba mostrando el cubrimiento de las funcionalidades probadas y los incidentes encontrados resultó ser una buena forma de comunicación del trabajo del equipo de pruebas.
- Sistema de seguimiento de incidentes web: En este caso se utilizó la herramienta Bugzilla, que permitió que el cliente pudiera validar los incidentes a medida que se iban reportando en la herramienta. Si bien el equipo de prueba le asignaba un tipo y criticidad a cada incidente, el cliente podía cambiar esto, si le parecía que no eran apropiados.

Los principales problemas que se presentaron durante el proyecto fueron de planificación, existiendo grandes diferencias entre lo planificado y el desarrollo del proyecto. Se encontraron dos problemas respecto de la planificación, por un lado problemas en las estimaciones realizadas por el equipo de prueba, que se deben principalmente a no contar con datos históricos y por otro lado los atrasos de

desarrollo debido a que la versión se entregó más tarde de lo planificado. Los atrasos en desarrollo, no dependen del equipo de prueba, pero debe negociarse desde el principio del proyecto como serán tratados los atrasos respecto al alcance del proyecto de prueba.

A continuación se describen los ajustes realizados a ProTest con el fin de mejorar la planificación y control del proyecto de prueba:

- Se agregaron elementos de PSP para la planificación y el seguimiento del proyecto. En particular se decide mejorar las mediciones para mejorar las estimaciones y con esto, mejorar la planificación del proyecto. A partir de estas mediciones se puede obtener un histórico de proyectos, de forma de poder mejorar las estimaciones en el futuro.
- Para mejorar las mediciones se debe mejorar la forma en que se reporta el esfuerzo invertido en las actividades del proyecto. Durante el caso de estudio, el reporte de esfuerzo se hizo por actividad ya que existen actividades que interesa saber sobre qué se estaba trabajando. Por ejemplo, para el reporte de incidentes, se registraron las horas insumidas en reportar incidentes, pero no se sabe cuánto tiempo llevó reportar cada incidente. En el caso de estudio se hubiera podido registrar el esfuerzo con mayor detalle, esto hubiera servido para la planificación de proyectos futuros. Se ajustó la actividad G2 – Reporte del Esfuerzo, en particular para las actividades interesa conocer:
  - P2 – Revisión de Requerimientos: tiempo utilizado por funcionalidad
  - D1-Diseño de los casos de prueba: esfuerzo por caso de prueba
  - E2 – Ejecutar las pruebas: tiempo de ejecutar cada caso de prueba
  - E3 – Testing exploratorio: tiempo por sesión de testing exploratorio
  - E4 – Reporte de Incidentes: esfuerzo de reportar cada incidente
  - E5 – Verificación de las correcciones: esfuerzo por caso de prueba de regresión.
- Las pruebas de regresión en el segundo ciclo llevaron mucho más tiempo del esperado, por lo que se debería tener una proporción del tiempo total del ciclo que se utilizará en ejecutar pruebas de regresión, la cual debe ser negociada con el cliente al principio del proyecto. A partir del cambio introducido en la forma de reportar el esfuerzo, se conocerá el tiempo que insume ejecutar cada caso de prueba en un ciclo. De esta forma, al comenzar cada ciclo de prueba, se puede conocer el tiempo que llevaría ejecutar las pruebas de regresión pedidas por el cliente. En caso de que el tiempo sea mayor del planificado, se debe negociar junto con el cliente qué pruebas volver a ejecutar.

Además de los cambios al proceso descritos previamente, se hicieron los siguientes ajustes al proceso luego de su puesta en práctica:

- Se realizaron diagramas de actividades para cada etapa del proceso. Con esto se busca explicar mejor la interacción entre las distintas actividades de una etapa.
- Se modificaron los nombres de algunas actividades, para reflejar mejor el objetivo que se busca con dicha actividad. También se cambiaron algunos entregables de entrada y salida de actividades y se mejoraron las descripciones de algunas actividades.
- Las actividades G1 –Estimación de Tareas y G2 – Reporte de Esfuerzo, que se encontraban solamente en la etapa Ciclo de Prueba, se agregaron en el resto de las etapas, ya que en todas las etapas se requiere estimar el tiempo que va a llevar realizar una actividad y luego, reportar el tiempo que realmente insumió realizarla.

Por último se destaca que las etapas de ProTest fueron seguidas según se definieron en el Capítulo 4. A partir de su puesta en práctica se valida que las etapas están bien definidas y modelan cómo debe desarrollarse el proyecto de prueba.

Capítulo 8

CONCLUSIONES

En este trabajo se estudia las técnicas, estrategias y metodologías para las pruebas de software. Se reseñan los principales procesos de prueba existentes y los modelos de calidad y mejora de procesos y productos de software. Se propone un proceso para la prueba funcional independiente de productos de software, llamado ProTest. El proceso es aplicado en un proyecto de prueba de una aplicación de gestión. Se presentan los resultados y los ajustes realizados a ProTest luego de su puesta en práctica. Para la documentación y administración de ProTest se utiliza la herramienta DeVeloPro.

El trabajo realizado puede ser resumido en tres resultados.

- La obtención de un resumen del estado del arte sobre las pruebas y el proceso de prueba.
- La definición de un proceso para la ejecución de pruebas de productos de software independiente.
- La documentación del proceso definido y la administración de los cambios al mismo en forma automática mediante la herramienta DeVeloPro.

8.1 Conocimiento sobre Pruebas y Proceso de Prueba

El proceso de desarrollo de software describe la vida de un producto de software desde su concepción hasta su entrega, utilización y mantenimiento. En forma análoga, el proceso de prueba describe la forma en que el producto de software debe ser probado.

El proceso de prueba puede ser visto como una parte del proceso de desarrollo de software o independiente de él. En éste último caso, el proceso de prueba no tiene en cuenta la forma en que se realiza el desarrollo para definir las actividades a realizar.

Se resume el estado del arte respecto a las pruebas de software. Se presentan las definiciones y objetivos de las pruebas y otros términos relacionados. Se describen las consideraciones a tener en cuenta al probar, se describen los distintos tipos de prueba: unitaria, de integración y del sistema y la distinción entre prueba funcional y estructural. Se presentan las técnicas de prueba haciendo énfasis en las técnicas de caja negra dado que son las usadas en ProTest, ya que es un proceso para pruebas funcionales. Se introduce la importancia de las mediciones en el proceso de pruebas y se discute sobre la independencia y el outsourcing de las pruebas del producto.

Se describe la relación entre proceso, ciclo de vida del software y proceso de desarrollo. Se estudian los antecedentes respecto a los procesos de prueba existentes en la literatura. Los distintos autores definen un ciclo de vida específico para las pruebas, que en general separa en distintas etapas las actividades de planificación, diseño y las de ejecución de las pruebas. Estos fueron los principales aportes para la elaboración del Proceso de Prueba que se propone en este trabajo.

Con el fin de definir el proceso de prueba en un marco de mejora continua, se estudian modelos de mejora de la calidad y los procesos, como el modelo de calidad ISO/IEC 9126, el enfoque Goal-Question-Metric, el modelo de mejora de procesos CMMI y el modelo de proceso personal PSP. Para la mejora específica del proceso de prueba se estudian Testing Maturity Model (TMM) y Test Process Improvement (TPI).

8.2 ProTest

Se define un proceso para realizar pruebas funcionales de un producto de software por parte de un equipo de pruebas independiente. El proceso fue creado para ser usado en el Laboratorio de Testing Funcional del Centro de Ensayos de Software.

El proceso de prueba definido es independiente del proceso seguido para desarrollar el producto, tiene como objetivo probar las funcionalidades del producto a partir de sus especificaciones, se basa en el análisis de riesgo del producto para definir las funcionalidades a probar y la prioridad con que serán probadas, organiza el proyecto de prueba en ciclos de prueba, que guían el proceso de prueba y acompañan el ciclo de vida del producto. ProTest cuenta de cuatro etapas: Estudio Preliminar, Planificación, Ciclo de Prueba y Evaluación del Proyecto.

Las etapas de ProTest son el resultado de fusionar las distintas propuestas discutidas en la sección 2.3.2, Fases de las Pruebas del Capítulo 2. Estas propuestas fueron adaptadas para ser usadas en una evaluación independiente. A diferencia de las otras propuestas, los ciclos de prueba en ProTest son lo que guían el proceso de prueba.

Cada ciclo de prueba está asociado a una versión del producto a probar. Durante la construcción de un producto, sin importar cual sea el proceso de desarrollo, se van generando distintas versiones de la aplicación. Las actividades de la prueba se realizan para una determinada versión del producto, sobre la cual se ejecutan las pruebas y se reportan los incidentes encontrados. Las pruebas que serán ejecutadas sobre una versión son planificadas con anticipación y deberían ser ejecutadas, a menos que las prioridades cambien. Con cada nueva versión del producto se realizan alguna o todas las tareas asociadas a las pruebas, a esto se le llama un ciclo de prueba.

Se definen las actividades a realizar en cada etapa de ProTest. Cada actividad se compone de un conjunto de artefactos de entrada que son las precondiciones para su ejecución, un conjunto de roles que la realizan que son los participantes de la misma, una descripción de las tareas a realizar en la actividad y un conjunto de artefactos de salida que son las poscondiciones de su ejecución.

Se presenta como caso de estudio la aplicación de ProTest en un proyecto de prueba de un sistema de gestión. Los principales problemas encontrados en el proyecto fueron de planificación, existiendo importantes diferencias entre lo planificado y lo que realmente sucedió. Al detectar este problema se mejoraron y ajustaron actividades en el proceso, con el objetivo de mejorar las estimaciones, mediciones y la planificación del proyecto de prueba. A pesar de los problemas de planificación, el caso de estudio valida que las etapas del proceso están bien definidas y modelan el desarrollo de un proyecto de prueba. El proceso resultó una guía útil para el equipo de prueba durante todo el proyecto y el cliente quedó conforme con la calidad de las pruebas y el resultado de las mismas.

Actualmente, el proceso es usado como metodología de trabajo en el Laboratorio de Testing Funcional del Centro de Ensayos de Software para la prueba de productos de software. También se ha utilizado como base en consultorías para el armado de laboratorios de testing dentro de las empresas de desarrollo de software.

8.3 Documentación y Ajustes del Proceso

Los procesos son llevados a cabo por personas, por lo que sus descripciones deben estar disponibles a todos los miembros de la organización. Se debe contar con documentación accesible del proceso, en un formato imprimible. Los cambios introducidos en el proceso requieren generar nuevamente la documentación, si el proceso tiene gran número de actividades, cada cambio puede resultar difícil de realizar, pudiendo quedar inconsistente la información en distintos elementos del proceso.

Al momento de comenzar esta tesis, no existían herramientas gratuitas que permitieran realizar esto. Se plantea por este motivo la necesidad de construirla. Se definieron los requerimientos que la herramienta debía cumplir y se realizó el seguimiento de su desarrollo cumpliendo el rol de cliente, validando los distintos prototipos generados de la herramienta. Un grupo de estudiantes desarrolló esta herramienta

La herramienta desarrollada se llama DeVeloPro es utilizada para generar la documentación de ProTest. En este trabajo es utilizada para mostrar las actividades en forma gráfica del Capítulo 5.

El sitio web de ProTest fue generado automáticamente con la herramienta DeVeloPro, también se generó el documento imprimible en formato pdf. La documentación generada resulta fácil de consultar y de realizarle cambios. En el Anexo A se describen las principales funcionalidades de DeVeloPro, su uso con ProTest y las mejoras posibles a realizar a la herramienta.

8.4 Trabajo a futuro

Hasta el momento, los servicios de prueba funcional independiente del CES han sido contratados por empresas Uruguayas desarrolladoras de software. En dichos proyectos de prueba se ha utilizado ProTest con buenos resultados. Resulta interesante probar el proceso, en un contexto donde el cliente no sea parte de la empresa que desarrolla el producto, sino que sea quien compra el producto. Si bien ProTest tiene en cuenta este contexto, podría requerir que algunas actividades deban ser ajustadas luego de su aplicación.

El Centro de Ensayos de Software cuenta con dos laboratorios, el Laboratorio de Testing Funcional, donde se realizan pruebas funcionales y el Laboratorio de Plataformas, donde se realizan pruebas del sistema referentes a requerimientos no funcionales como por ejemplo: desempeño, estrés, carga, entre otros. Muchos proyectos de prueba, requieren el trabajo conjunto de ambos laboratorios, el cliente puede necesitar realizar prueba funcional y prueba de desempeño de su producto. Como trabajo a futuro se encuentra definir un proceso común a ambos laboratorios. Dicho proceso tomaría ProTest y lo extendería teniendo en cuenta las actividades específicas del laboratorio de plataformas. Con este se logra un proceso común para todo el Centro de Ensayos, además de contar con plantillas unificadas para la entrega al cliente. Otra ventaja de esto, es que le pasaje de personal de una laboratorio a otro sería más fácil, ya que la terminología base sería la misma.

Con el objetivo de mejorar la planificación de los proyectos de prueba, en el futuro se espera contar con una cantidad importante de datos históricos resultado de las mediciones introducidas respecto al esfuerzo requerido para realizar las distintas actividades de ProTest. A partir de estos datos se espera poder mejorar las estimaciones de los proyectos de prueba.

Dentro de las extensiones a realizar a ProTest, se encuentra definir las actividades específicas a realizar en caso de que se quiera automatizar las pruebas funcionales utilizando herramientas. En la sección 2.4.3 del Capítulo 2 se describen las pruebas funcionales automatizadas. En la literatura, existen libros dedicados enteramente al proceso de pruebas automatizadas, este tema quedó fuera del alcance de este trabajo de tesis, pero resultaría muy útil para el Centro de Ensayos contar con las actividades específicas para realizar la automatización funcional en caso de ser necesario.

Otro punto a explorar, relacionado con lo anterior es el uso de herramientas que ayuden en todo el proceso de pruebas, no solamente aquellas específicas a las pruebas funcionales. En el caso de estudio se han utilizado herramientas para el seguimiento de incidentes y la organización de los casos de prueba, resulta interesante seguir explorando distintas herramientas que ayuden en las actividades del proceso de prueba definido.

BIBLIOGRAFÍA

[Bac99] Bach J. "Risk-based Testing", Software Testing and Quality Engineering Magazine Vol. 1, No. 6, November-December 1999.
http://www.stqemagazine.com/featured.asp?stamp=1129125440

[Bac00] Bach J. "Session Based Test Management", Software Testing and Quality Engineering Magazine Vol.2, No. 6, Noviembre 2000.
http://www.satisfice.com/articles/sbtm.pdf

[Bac01] Bach J. "What is Exploratory Testing? And How it differs from Scripted Testing" StickyMinds, Enero 2001.

[Bac03] Bach J. "Exploratory Testing Explained", The Test Practicioner, 2002.
http://www.satisfice.com/articles/et-article.pdf

[Bas88] Basili V., Caldiera G., Rombach D. "The Goal Question Metric Approach", Wiley& Sons Inc, 1994

[Bec99] Beck K. "Extreme Programming Explained, Embrace Change", ISBN 201-61641-6, Addison Wesley, 1999.

[Bei90] Beizer B. "Software testing techniques (2nd ed.)", ISBN:0-442-20672-0, Van Nostrand Reinhold Co, 1990.

[Bla02] Black R. "Managing the Testing Process, 2nd Edition". ISBN 0-471-22398-0, Editorial Wiley, 2002.

[Boe84] Boehm B. "Verifying and Validating Software Requirements and Design Specifications", IEEE Software, vol. 1, pp. 75-88, 1984.

[Boe88] Boehm B. "A spiral model of software development and enhancement", IEEE Computer; Volume 21, Issue 5, pp 61 – 72, May 1988.

[BS196] Burnstein I., Suwannasart T., Carlson C. "Developing a Testing Maturity Model: Part I", llinois Institute of Technology. Published in Crosstalk, August 1996.

[BS296] Burnstein I., Suwannasart T., Carlson C. "Developing a Testing Maturity Model: Part II", llinois Institute of Technology. Published in Crosstalk, September 1996.

[Bug06] Bugzilla, http://www.bugzilla.org/, 2006.

[Cas06] Castor, http://www.castor.org/, 2006.

[CES06] Centro de Ensayos de Software (CES) http://www.ces.com.uy, 2006.

[CMMI02] "Capability Maturity Model Integration (CMMI)", Version 1.1, CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1), Continuous Representation CMU/SEI-2002-TR-011 ESC-TR-2002-011, 2002.

[Dai94] Daich G., Price G., Raglund B., Dawood M., "Software Test Technologies Report", Test and Reengineering Tool Evaluation Project, Software Technology Support Center, 1994.

[Dev06] Aguirre A; Pais C; Tiscornia E; Baptista F; Toledo F; Muñoz H; Reina M; Moleri P; Ricca P; Patiño R; Montico S; De Uvarow S; "DeveloPro", Proyecto de Ingeniería de Software, Director: Raquel Abella, Cliente: Beatriz Pérez, Facultad de Ingeniería, Universidad de la República, 2005.

[Dij70] Dijkstra, E. "Notes on structures Programming", TH Report 70 –WSK-03, http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF, 1970.

[Doc06] DocTesting, http://www.qafactory.com/01doctesting.htm , 2006.

[DP06] Delgado A., Perez B. "Modelo de Desarrollo de Software OO" ISBN 970-94770-0-5, Proceedings de las V Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'06), Puebla, México, 1 de Febrero de 2006.

[DRP99] Dustin E., Rasca J., Paul J. "Automated Software Testing", ISBN 0-201-43287-0, Addison Wesley, 1999.

[DVP06] DeVeloPro, http://www.fing.edu.uy/~bperez/develoPro/, 2006.

[Fop06] Fop, http://xmlgraphics.apache.org/fop/, 2006

[GC06] Castro A., German M. "Evaluación de Arquitecturas de Software con ATAM (Architecture Tradeoff Analisis Method)", Director: Andrea Delgado, Pasantía , Facultad de Ingeniería, Universidad de la República, 2005.

[Gra06] Gareppe A., Perez D., Pastro J., Budelli E., Pittier A., Pirez G., Martinez H., Carrera R., Rodríguez J., Fradiletti J., Restuccia E. "Graphead", Proyecto de Ingeniería de Software, Director: Joaquín Goyoaga, Cliente: Beatriz Pérez, Facultad de Ingeniería, Universidad de la República, Uruguay, 2004.

[Het88] Hetzel B. "The complete guide to Software Testing, 2nd Edition", ISBN 0-89435-242-3, QED Information Sciences Inc., 1988.

[Hib06] Hibernate, http://www.hibernate.org/, 2006.

[Hir00] Hirschberg M. "The V Model", Crosstalk, June 2000.
http://www.stsc.hill.af.mil/crosstalk/2000/06/Hirschberg.html

[IEEE 610.12-1990]
IEEE Standard Glossary of Software Engineering Terminology Institute of Electrical and Electronics Engineers,ISBN: 155937067X, 1990.

[ISQ05] International Software Testing Qualifications Board, Certified Tester Foundation Level Syllabus, Versión 2005.
http://www.istqb.org/fileadmin/media/SyllabusFoundation.pdf

[Jav06] Sun Microsystems Java, http://www.sun.com/java/, 2006.

[JBR99] Jacobson I., Booch G., Rumbaugh J., "The Unified Software Development Process", Addison Wesley, 1999.

[JFC06] Java Foundation Classes (JFC), Swing packages, http://java.sun.com/docs/books/tutorial/uiswing/, 2006.

[JGr06] JGraph, http://www.jgraph.com/, 2006.

[JRE06] Sun Microsystems Java Runtime Environment Versión 1.5 (JRE)
http://java.sun.com/j2se/1.5.0/docs/relnotes/version-5.0.html, 2006.

[Kan00] Kaner C. "An Outline for Software Testing Outsourcing", Draft 6, revises the STAR draft , Marzo 2000.
http://www.kaner.com/pdfs/outsource.pdf

[Kan01] Kaner C. "Measurement Issues and Software Testing" , QUEST conference, Orlando FL., Marzo 2001.
http://www.kaner.com/pdfs/measurement_segue.pdf

[KBP01] Kaner C., Bach J., Pretichord B. "Lessons Learned in Software Testing", ISBN 0471081124, Wiley, 2001.

[KFN99] Kaner C., Falk J., Nguyen H. "Testing Computer Software, 2nd Edition", ISBN: 0471358460, Wiley, 1999 .

[Kit95] Kit E. "Software Testing In The Real World : Improving The Process", ISBN 0201877562, Addison Wesley, 1995.

[KoP98] Koomen T., Pol M. "Improvement of the Test Process using TPI", Sogeti, 1998.
http://www.uk.sogeti.com/services/PDF/TPI_flyer_21203.pdf

[Mye04] Myers G. "The art of software testing, 2nd edition", ISBN 0-471-46912-2, John Wiley & Sons Inc., 2004.

[Per06] Perez B. "ProTest – Proceso de Testing Funcional", ISBN 970-94770-0-5, Proceedings de las V Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'06), Puebla, México, 1 de Febrero de 2006.

[Pfl01] Pfleeger S. "Software Engineering, 2nd Edition", ISBN: 0130290491, Prentice Hill, 2001.

[PG06] PostgreSql, http://www.postgresql.org/, 2006.

[PMC05] Pomeroy-Huff M., Mullaney J, Cannon R., Sebern S. "The Personal Software Process (PSP) Body of Knowledge", Version 1.0., CMU/SEI-2005-SR-003 . http://www.sei.cmu.edu/publications/documents/05.reports/05sr003.html

[PTV02] Pol M., Teunissen R., van Veenendaal E., "Software Testing, A guide to the TMap Approach", ISBN: 0-201-74571-2, Addison Wesley, 2002.

[Sto03] Stolovich L.,"Análisis: Qué indican los datos de la industria uruguaya de tecnologías de la información", Cámara Uruguaya de Tecnologías de la Información (CUTI), Marzo 2003. http://www.cuti.org.uy

[SWE04] Guide to the Software Engineering Body of Knowledge SWEBOK, 2004 version. IEEE Computer Society. http://www.swebok.org, 2004.

[Swi06] Java Foundation Classes (JFC) Swing packages , Class JTree. http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JTree.html, 2006.

[Tri04] Triñanes, J. "Centro de Ensayos de Software" IV Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento, Noviembre 2004 http://is.ls.fi.upm.es/jiisic04/Cooperacion/sc1.pdf

[Whi00] Whittaker J., "What is Software Testing? And Why Is It So Hard?", IEEE Software, Vol. 17, No. 1, pp. 70-79, January 2000.

[Whi02] Whittaker, J. "How to break Software, a practical Guide", ISBN: 0201796198, Addison Wesley, 2002.

[Xal06] Xalan, http://xml.apache.org/xalan-j/, 2006.

Anexo A

DEVELOPRO

En este Capítulo se presenta DeVeloPro, herramienta para la documentación y administración de los procesos de una organización, permite definir los distintos elementos de un proceso como son: actividades, roles, artefactos, disciplinas, fases, iteraciones, definiciones y herramientas, la definición de las actividades se realiza en forma gráfica. Permite versionar estos elementos de forma de conocer qué cambio, por qué cambio y quién lo cambio.

Los procesos son llevados a cabo por personas, por lo que, deben estar disponibles a todos los miembros de la organización, es deseable contar con documentación accesible del proceso, en un formato imprimible. Los cambios introducidos en el proceso, requieren generar nuevamente la documentación, si el proceso tiene gran número de actividades, cada cambio puede resultar engorroso de realizar, pudiendo quedar inconsistente la información en distintos elementos del proceso. DeveloPro genera automáticamente la documentación en dos formatos: un sitio web con páginas html estáticas y un documento en pdf, para cada versión del proceso.

En la sección A.1 se describen los principales requerimientos para DeveloPro, en la sección A.2 se presenta la tecnología utilizada para su desarrollo, en la sección A.3 se muestra el uso de DeveloPro con ProTest, en la sección A.4 se describen las conclusiones de su uso y por último en la sección A.6 se describe el trabajo a futuro con DeveloPro.

A.1 Requerimientos para DeveloPro

En esta sección se describen los principales requerimientos definidos para la herramienta. Los 7requerimientos que se detallan en esta sección son un resumen del Documento de Requerimientos que fue realizado por el grupo de estudiantes que realizó el producto. El documento completo puede ser consultado en [DVP05] donde se encuentra también la versión final del Modelo de Casos de Uso.

Desde un principio, se definió que la herramienta debía poderse utilizar para distintos tipos de procesos vinculados con un producto de software. El proceso de desarrollo de software es la estructura más general, ya que si se puede definir en la herramienta un proceso de desarrollo de software, se podrá definir también un proceso de testing.

A.2.1 Definición y Administración de Procesos

Un proceso para el desarrollo de software es un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a producir nuevo software. La herramienta debe manejar información de los procesos, mantener su nombre y descripción, permitir crear nuevos procesos, borrar, modificar o consultar los existentes.

Los elementos de un proceso que se deben administrar en la herramienta son los que se muestran en la Figura 54 y se describen a continuación. La herramienta permitirá la creación, modificación, borrado y consulta de elementos de un proceso.

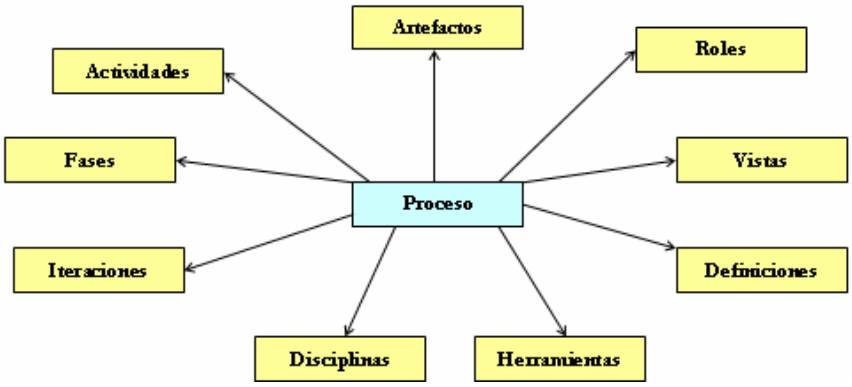

Figura 54– Elementos de un proceso

- **Actividades**: Una actividad es un conjunto de acciones que se realizan con el objetivo de crear o actualizar uno o varios artefactos. En la Figura 55 se muestra un esquema de los componentes de una actividad, cada actividad se compone de un conjunto de artefactos de entrada que son las pre-condiciones para su ejecución, un conjunto de roles que la realizan que son los participantes de la misma, una descripción de las tareas a realizar en la actividad y un conjunto de artefactos de salida que son las post-condiciones de su ejecución. Existe además, un rol que es el responsable de que la actividad se realice, definiciones que ayudan a entender la actividad y herramientas que asisten en la realización de la misma.

- **Artefacto**: Un artefacto es todo tipo de información creada, producida, cambiada o usada por el proceso, es un producto del proceso. Los artefactos pueden tener asociado una plantilla, como

guía de su contenido. Los artefactos podrán asociarse a las actividades como entrada o salida de las mismas.

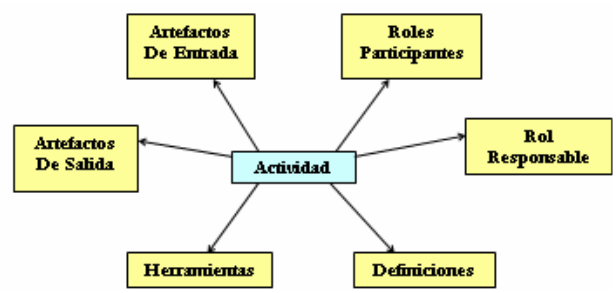

Figura 55– Actividad y sus relaciones

- **Disciplinas**: Un disciplina permite realizar un agrupamiento lógico de las diferentes actividades.

- **Fases e Iteraciones**: Los procesos de desarrollo de software dividen el desarrollo en ciclos donde cada ciclo trabaja para construir una nueva generación del sistema y a su vez cada ciclo se divide en fases y las fases en iteraciones. Cada fase tiene un conjunto de objetivos definidos y solamente se puede dar por concluida al alcanzar estos objetivos. Una Iteración es una recorrida completa a través de las disciplinas, realizando una mini-cascada en las actividades definidas, resultando en una liberación (interna o externa) de un ejecutable, el cual es un subconjunto del sistema final que crece incrementalmente de iteración en iteración hasta llegar al sistema final. Se permitirá la creación, modificación, borrado y consulta de fases e iteraciones

- **Rol**: Un rol define el comportamiento y responsabilidades de un individuo, o un conjunto de individuos que trabajan juntos como un equipo. Un integrante del proyecto puede cumplir varios roles así como también, un rol puede ser asumido por más de un integrante. Los roles tienen un conjunto de actividades cohesivas que realizan. Estas actividades están estrechamente relacionadas y funcionalmente acopladas, y se realizan mejor por el mismo individuo. Se podrá crear la asociación entre un rol y una actividad, está relación indica qué roles serán responsables de la actividad o qué roles participaran en la misma. Los roles podrán estar asociados a artefactos, esta relación representa que el rol es responsable por la creación del artefacto.

- **Herramientas**: Las herramientas asisten en la realización de las diversas actividades del proceso. Las herramientas podrán ser asociadas o desasociadas a una actividad. Esto indica que para realizar la actividad se usa la herramienta.

- **Definiciones**: Las definiciones son utilizadas para aclarar algún concepto importante referido a un elemento del proceso.

- **Vistas**: Es una forma gráfica de ver las precedencias que existen entre los elementos del proceso. El usuario podrá mediante una herramienta gráfica dar de alta, modificar, borrar y consultar una vista.

Es de interés poder referenciar elementos de un proceso en otro, esto implica que un elemento de un proceso, por ejemplo una actividad, pueda ser la misma entre dos procesos, creándose en uno de los procesos y se referencia en el otro. No sólo se hace referencia al nombre y la descripción de la actividad sino también todos los elementos con los que está relacionada como son entregables y roles, entre otros.

A.2.2 Versionado

Tanto el proceso como sus elementos, sufren modificaciones a medida que pasa el tiempo. Para llevar un control de las modificaciones, se crean versiones, que son un registro del estado del elemento en determinado momento. Cuando se realiza el versionado de un proceso se genera para el mismo una línea base con los elementos que tiene asociado el proceso en sus últimas versiones; luego de generar una versión de un proceso esta quedará almacenada en el sistema y no podrá modificarse. El usuario podrá realizar una nueva versión de un elemento manteniendo las versiones anteriores. Al realizar el versionado de un elemento en un proceso, se tomara la nueva versión para todos los elementos que se relacionen con ese elemento en ese proceso. Debe ser posible acceder a las versiones anteriores de un elemento en un proceso. Una versión de un elemento de un proceso puede estar en dos estados: Abierta, quiere decir que el usuario está realizando cambios sobre la versión y Cerrada, que es accedida solo para lectura.

Se podrá consultar cuáles son las diferencias entre dos versiones de elementos de un proceso, devolviendo un listado como resultado, donde se indica para cada cambio: qué cambio, por qué cambio y quién lo cambio.

Cuando se realiza el versionado de un proceso puede darse el caso de que algún elemento haya sido modificado pero no versionado, con lo cual el sistema mostrara todos los elementos en esta situación y permitirá al usuario elegir cuales desea versionar antes de realizar el versionado del proceso.

A.2.3 Herencia entre procesos

Los procesos pueden relacionarse entre sí en forma de herencia. Debe ser posible definir un proceso como instancia de otro, el nuevo proceso comenzará a desarrollarse a partir de lo especificado en el proceso padre, sobre lo cual luego se harán modificaciones y agregados que terminarán de definir este nuevo proceso. Esto significa que el proceso hijo tiene (además de las propias) todas las actividades del proceso padre, de forma que una modificación en el proceso padre impacta en el hijo.
Las funcionalidades de Herencia entre procesos y versionado están relacionadas, ya que si el proceso en el cual se hará el versionado del elemento tiene procesos hijos, cuando se haga el versionado hay dos posibles efectos sobre los hijos:

1. El proceso hijo no cambio el elemento para el cual se creo una nueva versión, con lo cual se realiza el cambio de versión en el proceso hijo también sin que este sea notificado del cambio de versión del elemento.
2. El proceso hijo si cambio el elemento que se va a versionar, entonces en este caso a la hora de cargar el proceso hijo se notificara lo sucedido en el proceso padre y se dará la posibilidad de cambiar a la nueva versión generada por el padre o mantener la actual.

A.2.4 Visualización gráfica

Interesa poder visualizar en forma gráfica el diseño de procesos, agregar actividades creando relaciones de precedencia y asociar estas actividades a fases del proceso, asociar roles, artefactos de entrada y salida; y otros elementos. También definir precedencias entre actividades en forma gráfica.

A.2.5 Generación de la documentación automática del proceso

La herramienta generará en forma automática el sitio web del proceso y el documento en formato pdf del mismo. Para este requerimiento, se planteó desde un principio la utilización de algún producto con

licencia GNU que realizara el pasaje automático de los elementos del proceso, representados en un archivo de intercambio de información, como por ejemplo XML a los formatos web y pdf.

A.2 Desarrollo de DeVeloPro

Se estudiaron las herramientas gratuitas existentes en el mercado que brindaran las funcionalidades descritas en la sección anterior. No se encontraron productos que cumplieran con las especificaciones buscadas, por lo que se decidió construir esta herramienta.

En el año 2004, un grupo de 12 estudiantes construyeron la herramienta Graphead [Gra04]. Dicha herramienta cuenta con una interfase muy amigable, pero tiene grandes problemas de desempeño, lo que hace imposible su uso. Durante el primer semestre del año 2005 un grupo de estudiantes realizando una pasantía evaluaron la arquitectura del producto, siguiendo el método ATAM, para conocer el origen de los problemas de desempeño [ATA05]. En el segundo semestre del año 2005, otro grupo de 12 estudiantes se les planteó mejorar el producto Graphead realizado en el año anterior, este grupo mejoró el producto, teniendo en cuenta las recomendaciones dadas por el grupo que evaluó la arquitectura de Graphead. A la nueva versión se le puso el nombre de DeVeloPro [DeV05]. En los tres casos, se cumplió el rol de cliente, definiendo los requerimientos de la herramienta y validando los distintos prototipos generados.

La documentación generada durante el proyecto para desarrollar DeVeloPro puede ser consultada en [DVP05]. A continuación se describe la tecnología utilizada para desarrollarlo.

La herramienta se desarrolló usando el lenguaje Java, se utilizó JDK versión 5.0 [JDK06] y para que la interfaz gráfica fuera más amigable se utilizaron los componentes de Java: Swing [Swi06], Jtree [JTr06] y JGraph [JGr06] para la visualización de diagramas.

Para la persistencia se utilizó el manejador de Base de Datos PostGreSQL 8.0.3 [PG06] y para realizar la correspondencia entre los objetos y las tablas relacionales se utilizó Hibernate 3 [Hib06].

Para generar el sitio web en forma automática, primero se realizó la correspondencia entre los objetos y un archivo XML con Castor en su versión 0.9.7 [Cas06], y la herramienta Xalan [Xal06] se utilizó para general el sitio web a partir del archivo XML y el formato dado por un archivo XSLT. Para generar el archivo PDF a partir del archivo XML, se utilizó Fop [Fop06].

A.3 Aplicación de DeveloPro con ProTest

En esta sección se describe el uso de la herramienta DeveloPro con el proceso ProTest. Se muestran las principales funcionalidades de DeveloPro al ingresar el proceso a la herramienta y el sitio web y el pdf generados automáticamente. Se encuentra disponible en [DVP06] el instalador de DeveloPro para descargar, así como el Manual de Instalación y el Manual Técnico del producto.

En la Figura 56 se muestra la organización de un proceso en la herramienta DeveloPro, cada carpeta agrupa los principales elementos del proceso que fueron definidos en la sección A.1.1. Se puede ver la barra de herramientas con las funcionalidades principales:

- Crear: Permite crear un nuevo proceso, Abrir: permite abrir un proceso ya existente,
- Cerrar: Permite cerrar un proceso abierto
- Web: Permite generar el sitio web del proceso
- Versionar: Permite versionar el proceso y/o un elemento del mismo
- Exportar: Permite exportar el proceso para ser usado en otra instalación de DeveloPro

- Importar: Permite importar un proceso desde otra instalación de DeveloPro
- Listar Diferencias: Muestra las diferencias entre dos versiones de un proceso o entre dos versiones de un elemento de un proceso.

Figura 56 – Proceso cargado

En la Figura 57 puede verse el editor gráfico de actividades, donde se muestra la actividad P2- Revisión de Requerimientos, que tiene como artefactos de entrada a FR- Fuentes de requerimientos e IP- Inventario de Prueba y como artefacto de salida a IP- Inventario de Prueba, los roles participantes en esta actividad son el Diseñador, Líder de Proyecto y el Cliente. El Diseñador es el responsable de la actividad, por eso la flecha es de color rojo. En el panel de la izquierda pueden verse las actividades definidas para el proceso.

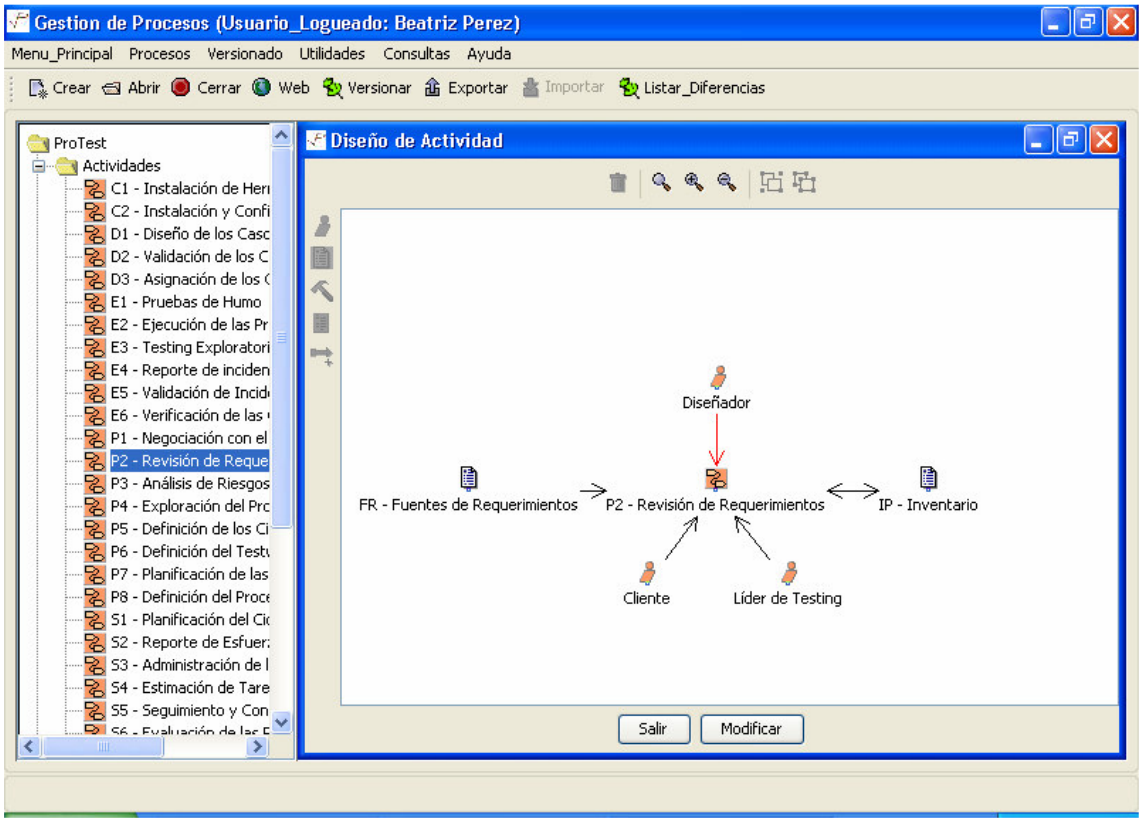

Figura 57– Editor gráfico de Actividades

Si se quiere versionar el proceso, con esta nueva actividad, se presiona el botón "Versionar", se muestran los elementos a versionar, como se puede ver en la Figura 58. Se puede versionar todo o seleccionar cuales versionar y cuales no. Al versionar permite incluir una descripción del motivo del cambio.

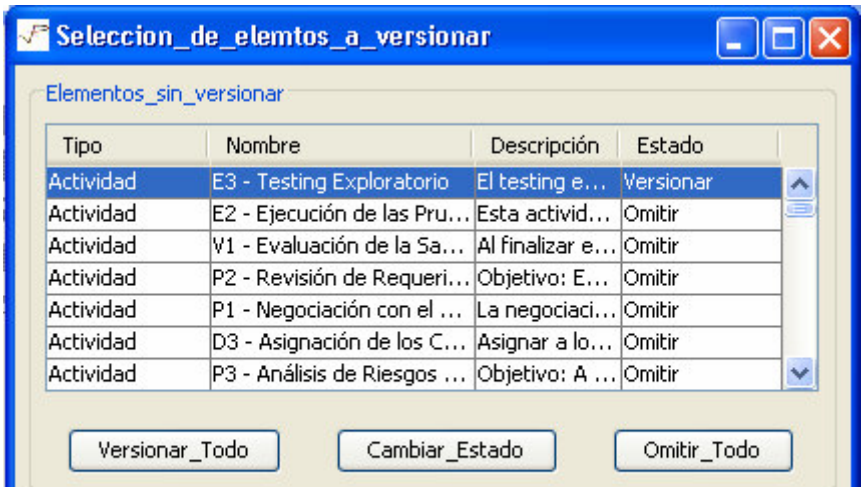

Figura 58– Elementos a Versionar

Una vez versionado el elemento, pueden consultarse las diferencias entre dos versiones, mostrando las características que han cambiado de una versión a otra. La lista de diferencias se puede obtener para dos

versiones de un elemento o para dos versiones de un proceso, en dicho caso muestra todos los elementos que cambiaron entre las dos versiones del proceso, esto se muestra en la Figura 59, con los botones anterior y siguiente se pueden recorrer las diferencias para el proceso.

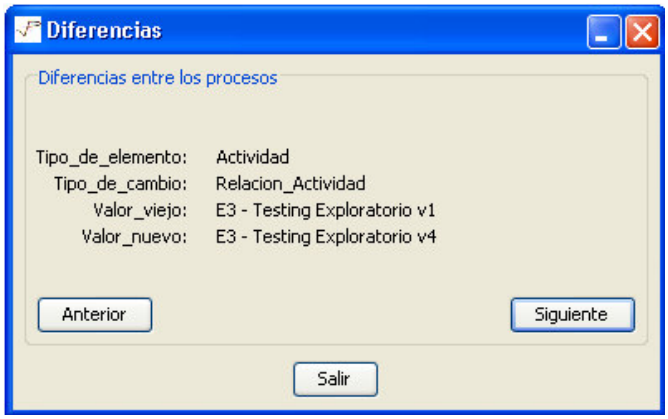

Figura 59- Diferencias entre versiones de un proceso

En la Figura 60 puede verse el editor de vistas, mostrando la precedencia entre actividades de la etapa de Diseño de las Pruebas. Esto resulta útil para explicar el flujo de las actividades que ocurren en una etapa.

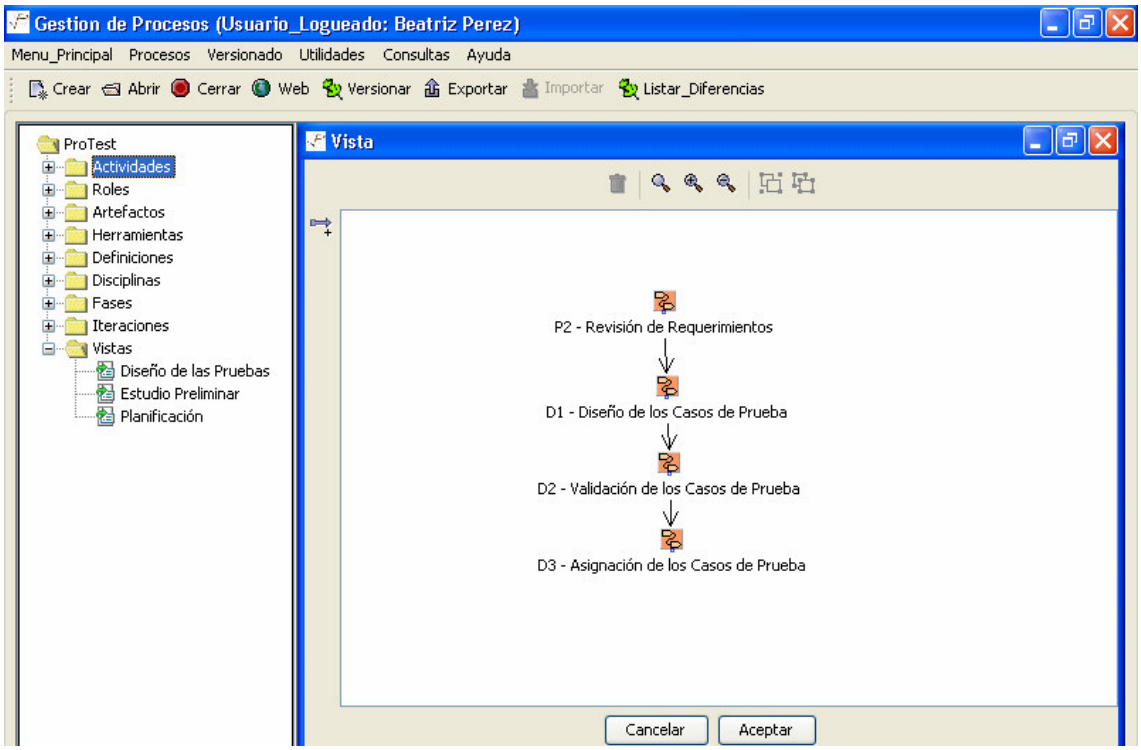

Figura 60– Editor de Vistas

Por último en la Figura 61 se puede ver el sitio web del proceso generado automáticamente, donde se muestra la pagina web de la actividad E5 – Reporte de Incidentes, donde la vista gráfica de la actividad aparece como un dibujo, si se presiona sobre alguno de los elementos del dibujo, por ejemplo, en el rol Analista, se activa el link a la página donde se explican las tareas de dicho rol.

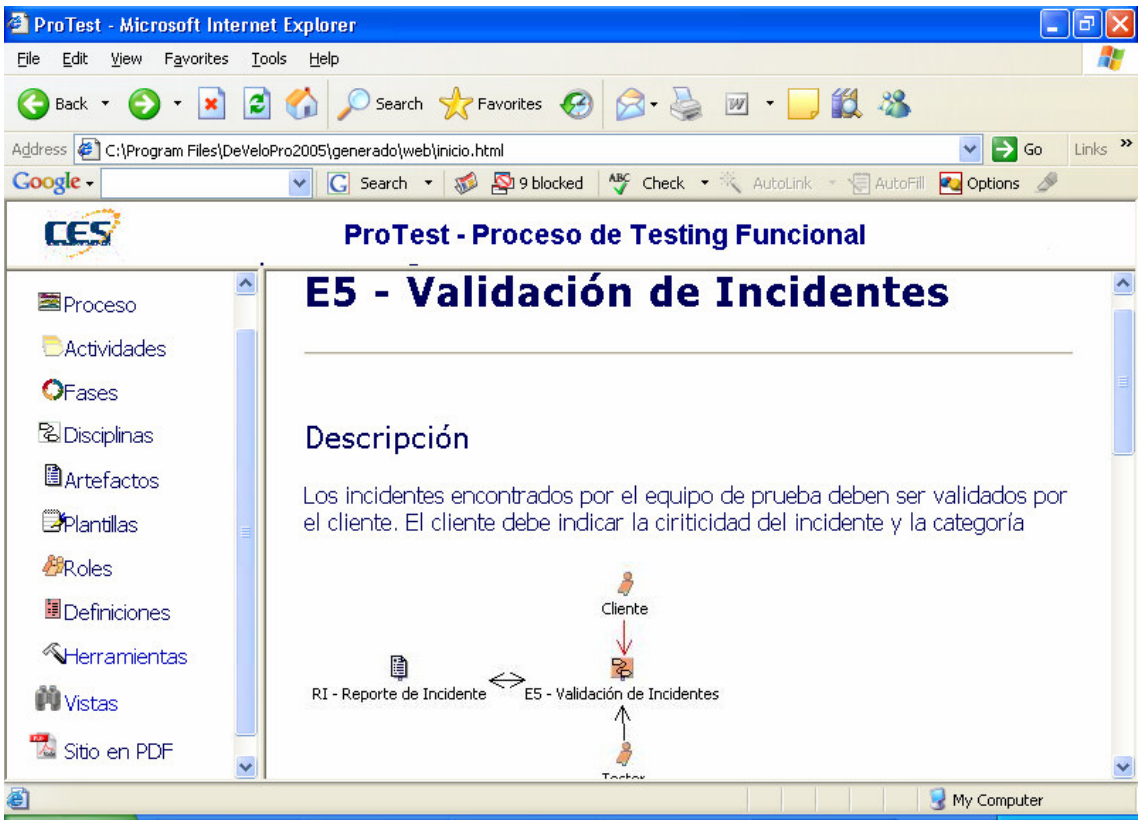

Figura 61– Sitio Web generado automáticamente por DeveloPro

A.4 Conclusiones del uso de DeveloPro

El sitio web generado automáticamente resulta agradable a la vista, permitiendo al usuario una navegación fácil y rápida. Durante la definición de ProTest , se han ingresado a la herramienta más de quince versiones del proceso, la herramienta ha demostrado trabajar bien con cualquiera de dichas versiones, pudiendo volver a cualquier versión definida anteriormente del proceso.

Como limitaciones de la herramienta DeveloPro se encuentran, que las descripciones de los elementos del proceso son ingresadas como texto plano, esto no permite realizar un tratamiento apropiado de algunos elementos, donde el formato del texto permite explicarlo mejor, mediante el uso, por ejemplo, de negritas o viñetas. La herramienta tampoco permite agregar información adicional a los elementos, como por ejemplo, figuras ilustrativas.

A.5 Trabajo a futuro con DeVeloPro

Como trabajo a futuro se encuentra mejorar la herramienta DeveloPro en los siguientes aspectos:

- Incluir un editor de texto en la herramienta donde se pueda dar formato a las descripciones de los elementos, como por ejemplo, viñetas, negritas, etc.
- Permitir el ingreso de información adicional en los elementos, como por ejemplo imágenes ó links.
- Mejorar el editor de vistas, de forma que se puedan realizar diagramas de actividades a partir de los elementos del proceso.
- Permitir personalizar los elementos que se tendrán en cuenta al generar el proceso, por ejemplo que se pueda decidir si se tendrán disciplinas e iteraciones.
- Generar reportes de los elementos que componen un proceso, por ejemplo, resulta interesante tener un listado de todas las actividades o un listado de las actividades en que participa un rol en particular.
- Complementar esta herramienta con otra que permita el seguimiento de los proyectos que siguen los procesos definidos por DeveloPro, permitiendo ingresar los artefactos correspondientes a las actividades, las horas que insumió cada actividad, la agenda real seguida y poder comparar el proceso ideal con el proceso real.