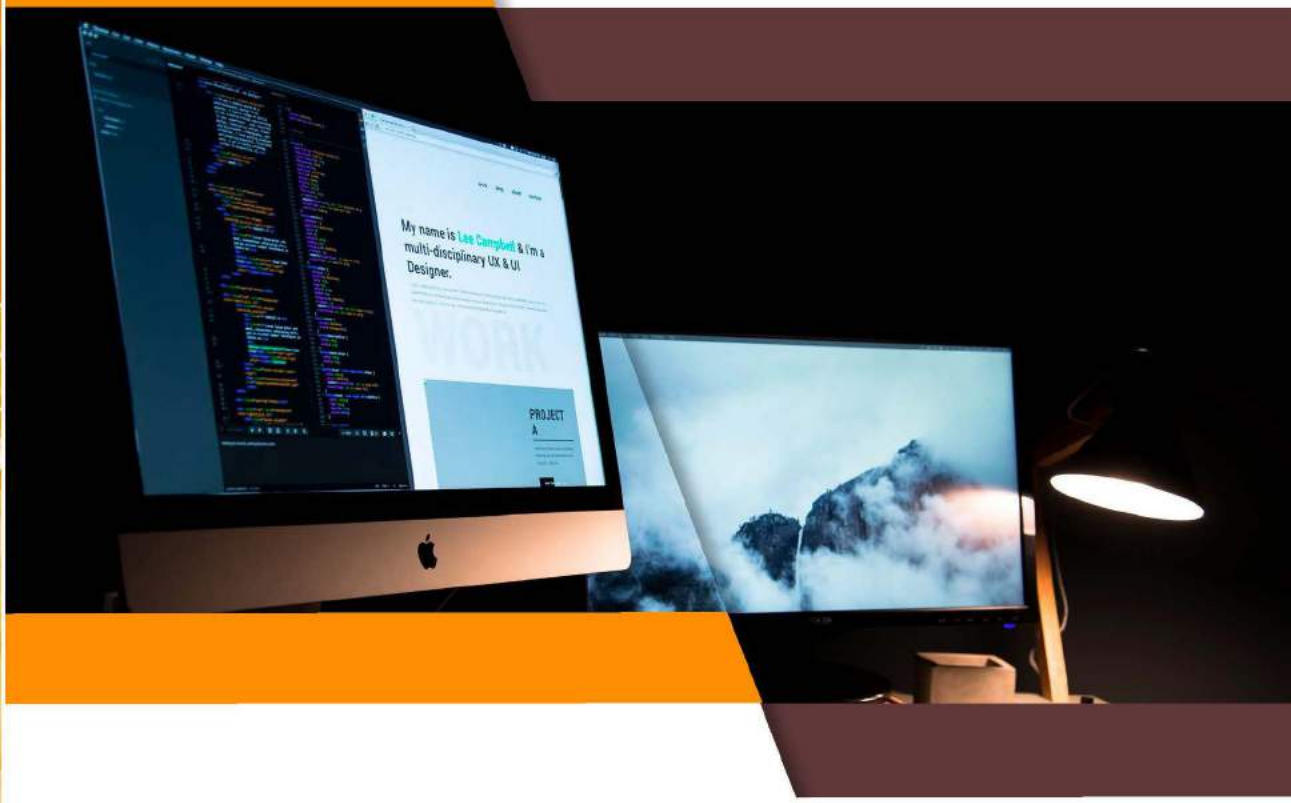
Calidad y pruebas en el desarrollo de Software

Dr. Jorge Cervantes Ojeda
Dra. María del Carmen Gómez Fuentes



Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

Obra ganadora del Tercer Concurso para la publicación de libros de texto y materiales de apoyo a la impartición de los programas de estudio de las licenciaturas que ofrece la Unidad Cuajimalpa

Calidad y Pruebas en el desarrollo de Software

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

1) Calidad y pruebas en el desarrollo de software
Clasificación Dewey: 005.12 C47
Clasificación LC: QA76.76.D47 C47
Cervantes Ojeda, Jorge
Calidad y pruebas en el desarrollo de software / Jorge Cervantes Ojeda, María del Carmen Gómez Fuentes . – Ciudad de México : UAM, Unidad Cuajimalpa, 2017.
107 p. : il. col., diagrs. ; 17 x 24 cm.
ISBN: 978-607-28-1075-4

1.Software informático – Desarrollo – Libros de texto. 2. Software informático – Control de calidad – Libros de texto. 3. Programación de computadoras – Libros de texto. 4. Informática – Manuales de laboratorio. 5. Universidad Autónoma Metropolitana – Unidad Cuajimalpa – Planes de estudio. I.Gómez Fuentes, María del Carmen, coaut.

Esta obra fue dictaminada positivamente por pares académicos mediante el sistema doble ciego y evaluada para su publicación por el Consejo Editorial de la UAM Unidad Cuajimalpa.

Avenida Vasco de Quiroga 4871
Col. Santa Fe Cuajimalpa, delegación Cuajimalpa de Morelos
C.P. 05348, Ciudad de México (Tel: 5814 6500)
www.cua.uam.mx

ISBN: 978-607-28-1076-1

Primera edición: 2017

Corrección de estilo: Adriana Rivera
Diseño editorial y portada: Literatura y Alternativas en Servicios Editoriales S.C.

Avenida Universidad 1815-c, Depto. 205, Colonia Oxtopulco,
C. P. 04318, Delegación Coyoacán, Ciudad de México.
RFC: LAS1008162Z1

Impreso y hecho en México
Printed and made in Mexico

Dr. Jorge Cervantes Ojeda y Dra. María del Carmen Gómez Fuentes

Calidad y Pruebas en el desarrollo de Software

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Dr. Eduardo Peñalosa Castro
Rector General

Dr. José Antonio De los Reyes Heredia
Secretario General

Dr. Rodolfo Suárez Molnar
Rector de la Unidad Cuajimalpa

Dr. Álvaro Peláez Cedrés
Secretario de la Unidad Cuajimalpa

Mtro. Octavio Mercado González
Director de la División de Ciencias de la Comunicación y Diseño

Dr. Raúl Roydeen García Aguilar
Secretario Académico de la División de Ciencias de la Comunicación y Diseño

Dr. A. Mauricio Sales Cruz
Director de la División de Ciencias Naturales e Ingeniería

Dr. José Javier Valencia López
Secretario Académico de la División de Ciencias Naturales e Ingeniería

Dr. Roger Mario Barbosa Cruz
Director de la División de Ciencias Sociales y Humanidades

Dr. Jorge Lionel Galindo Monteagudo
Secretario Académico de la División de Ciencias Sociales y Humanidades

Índice

1. Introducción

1.1 Presentación

El curso de Calidad y Pruebas en la UAM Cuajimalpa aborda los principios fundamentales de *calidad* y realización de *pruebas* en sistemas de Software. Ambos temas se enfrentan al mismo reto: la dificultad de que los alumnos realicen prácticas, en un laboratorio escolar, en las cuales apliquen los conceptos que se aprenden en clase. Considerando las características del Modelo Educativo de la UAM Cuajimalpa, este libro incorpora, como actividad integradora a la exposición de los temas teóricos, una serie de prácticas diseñadas para ejecutarse en una herramienta de apoyo novedosa llamada QualiTeam[1]. QualiTeam es una aplicación web desarrollada en la UAM Cuajimalpa que ayuda al control de la documentación asociada a proyectos de desarrollo de software siguiendo los principios y procedimientos del aseguramiento de la calidad de la Ingeniería del Software. Incluye un procedimiento de revisión de documentos entre colegas y la administración de los documentos de pruebas de acuerdo con el estándar IEEE-829[2], el estándar que se aplica actualmente en la industria. Con QualiTeam, es posible llevar a la práctica varios de los procedimientos de aseguramiento de la calidad, lo que ayuda a los alumnos a asimilar más fácilmente y mejor los conceptos expuestos en este libro. Además de la administración y revisión de documentos de un proyecto, esta herramienta facilita las actividades de desarrollo y mantenimiento (solicitudes de cambio y reportes de error) de un sistema de Software posterior a su entrega. El objetivo es que profesores y alumnos de cualquier licenciatura relacionada con la ingeniería de Software puedan utilizarlo para coordinar los trabajos en equipo de sus proyectos.

Los alumnos regulares cursan la UEA Calidad y Pruebas al mismo tiempo que Análisis de requerimientos. Los temas que se estudian en este libro complementan de manera transversal esta segunda UEA, ya que proporcionan los elementos para mejorar la calidad de los documentos de *Especificación de Requerimientos.*

1 Gómez Fuentes, M.C., Nolasco Cardiel L.A., Cervantes Ojeda, J. *QualiTeam: una herramienta de apoyo para el aprendizaje de Calidad y Pruebas del SW,* Avances recientes en Ciencias Computacionales- CIComp, 2016, pp. 46-53.

2 IEEE Standard for Software Test 829-1998. *IEEE Computer Society*, *345*, 10017-2394.

1.2 El programa de estudios de "Calidad y Pruebas"

A continuación presentamos el contenido sintético del programa de estudios de la UEA Calidad y Pruebas.

Contenido Sintético

1. Garantía de la calidad.
   - Conceptos básicos sobre la calidad y la calidad del software.
   - Organización de la calidad del software.
   - Actividades para el control y la garantía de la calidad del software.
   - Estándares y normas.
   - Medida de la calidad.
   - Planes de la calidad.
2. El proceso de pruebas de software y la metodología de desarrollo.
   - El desarrollo de proyectos y el ciclo de desarrollo de pruebas.
   - Captación de requisitos.
   - Análisis funcional.
   - Diseño técnico.
   - Construcción.
3. Roles en el proceso de las pruebas de software.
   - El equipo de pruebas.
   - Consultor de pruebas.
   - Ingeniero de pruebas.
4. Tipos de pruebas.
   - Pruebas unitarias.
   - Pruebas de integración.
   - Pruebas de sistemas.
   - Pruebas de aceptación.
5. Productos y herramientas de pruebas de software.
   - Productos comerciales.
   - Productos de código abierto.

Como se puede apreciar en el contenido, este libro cubre el 100% del programa de estudios Calidad y Pruebas de la UEA, y, adicionalmente, proporciona material para realizar prácticas de laboratorio en la herramienta Qualiteam.

1.3 Objetivo

1.3.1 Objetivo General

Conocer y aplicar las técnicas y procedimientos básicos de un sistema de aseguramiento de la calidad del software.

El objetivo general del Plan de Estudios de Ingeniería en Computación es el siguiente:

> Formar profesionales críticos, creativos, con responsabilidad social y compromiso ético, calificados en tecnologías avanzadas, así como en el diseño y *la producción de software de calidad*, con capacidad para trabajar en equipos multidisciplinarios, así como habilitados para identificar y resolver problemas de forma innovadora, en diversas áreas de la tecnología.[3]

En este libro se estudian las técnicas más conocidas para lograr la creación de Software con la mínima cantidad posible de defectos. Esto capacita a los alumnos para llevar a cabo con calidad, los procesos de desarrollo de software y su mantenimiento, tal y como se plantea en el objetivo general del Plan de Estudios. Además, los temas aquí expuestos cubren el tercero de los objetivos específicos del Plan, que es: "Aplicar los estándares de calidad para la construcción de sistemas de software".

1.4 Conocimientos, habilidades y actitudes

A continuación se explica por qué es importante este libro de texto para los alumnos de Ingeniería en Computación y de qué manera concuerda con los objetivos del Plan de Estudios.

1.4.1 Los conocimientos a adquirir y su importancia

Los sistemas de Software se utilizan ampliamente en la vida moderna. Se les usa en la ingeniería, en la ciencia, en la administración de empresas, en la medicina, en la economía, en el entretenimiento, etc. Con la rápida expansión del uso de computadoras hacia toda la sociedad y con el consecuente crecimiento del tamaño de los programas, el control de calidad se ha vuelto un aspecto fundamental del desarrollo de Software. Por otra parte, la fase de pruebas es indispensable en todo proyecto de Software. Ningún producto con un mínimo grado de complejidad está exento de defectos que, al menos en potencia, generan *fallas*. Dado que el

3 http://www.cua.uam.mx/estudiar-en-la-uam-cuajimalpa/licenciaturas/ingenieria-en-computacion

Software se usa cada vez más en aparatos en los que las consecuencias de las *fallas* son sumamente graves (muerte, fracaso en misiones importantes, quebrantos de seguridad, pérdidas financieras o sociales), ya no es viable entregar un producto sin haber pasado por un proceso de pruebas que proporcione los niveles de confianza necesarios para poder usarlo.

1.4.2 Las habilidades y actitudes a desarrollar

Las habilidades que deberá adquirir el alumno a lo largo de este curso son:

- *Lenguaje disciplinar*: aprenderá el lenguaje del aseguramiento de la calidad del Software y del desarrollo de pruebas.
- *El trabajo armónico en equipo*: al poner en práctica el proceso de revisión entre colegas los integrantes deberán aportar mejoras al producto, teniendo en cuenta la cordialidad y el respeto.
- *Comunicación eficaz de forma oral y escrita*: en las prácticas trabajarán con documentos elaborados por los alumnos. El alumno deberá ser capaz de escribir documentos completos y de hacer crítica constructiva cuando escriba sus comentarios en los documentos de sus compañeros.

2. Calidad del Software

2.1 Objetivos específicos del capítulo

- Comprender la importancia de la gestión de la calidad en la Ingeniería de Software.
- Explicar qué es el aseguramiento de la calidad del software.
- Comprender la utilización del estándar IEEE 730 para la creación de un plan de aseguramiento de la calidad del software.
- Comprender el procedimiento de revisión entre colegas de productos de software.
- Comprender el procedimiento de inspección de productos de software.
- Comprender la elaboración de la documentación de pruebas de software según el estándar IEEE 829.

2.2 Conceptos básicos sobre la calidad del software

2.2.1 El concepto de calidad de un producto

Para aumentar la competitividad de las empresas a finales del siglo XIX, se comenzó a garantizar que los productos que genera una fábrica no tuvieran defectos. La garantía implica ofrecer al cliente que en caso de que su producto presente algún defecto, se le substituirá por otro.

Como la substitución de productos defectuosos tiene un costo adicional de operación que no puede cobrársele al cliente y debe ser absorbido por la empresa, entonces, para evitar que los productos que llegan al cliente tengan defectos se comenzó a aplicar un proceso de *control de calidad* en las fábricas. Este control de calidad se limita a la verificación de que los productos que salen de la línea de producción no tengan defectos y, en caso de tenerlos, se les separa para asegurar que no sean vendidos.

El siguiente paso fue la generalización del concepto de productos de calidad a *compañía de calidad* en el que se pretende que todas las personas que trabajan en la compañía se hagan res-

ponsables de la calidad de su trabajo, es decir, hacer un cambio de fondo en la cultura de la organización. Aún con los esfuerzos anteriores, fue necesario hacer más, ya que era indispensable tomar en cuenta al consumidor. En ocasiones las ventas podían bajar a pesar de haber hecho un "buen producto". Para mejorar esto fue necesario especificar los *requerimientos* del cliente y, en base a éstos, hacer las especificaciones técnicas del producto, planes, manufactura y control de calidad. Las quejas de los clientes debían ser atendidas y resueltas al 100% y consideradas para el futuro. La documentación debía ser completa y adecuada.

Los conceptos anteriores de calidad ya estaban funcionando en otras industrias cuando surgió la industria del Software. Al inicio de la era de la computación, había muy pocos usuarios de Software y los programas no eran muy grandes, así que no se vislumbraba la necesidad de mejorar la calidad del SW. Simplemente se volvía a hacer y ya. En la actualidad el Software tiene aplicaciones en casi en todas partes y en todos los aspectos de la vida actual. Con la rápida expansión del uso de computadoras hacia toda la sociedad y con el consecuente crecimiento del tamaño de los programas, la calidad se volvió una parte fundamental del desarrollo de SW.

A diferencia de otras industrias, la del Software tiene las dos propiedades siguientes:

- Es muy fácil hacer modificaciones en el diseño del producto.
- La producción masiva de un producto prácticamente no produce *fallas*, porque consiste en copiar archivos.

Entonces, ¿por qué hay *fallas*?

La causa principal es precisamente la facilidad para hacer modificaciones. Los autores de esta obra consideramos que cuando un diseñador de Software hace un programa, se puede ver tentado a esperar a que las pruebas le indiquen si su diseño fue exitoso o no. De esta manera, si una prueba *falla*, entonces haría alguna modificación para corregirlo, ya que "editar y compilar no cuesta mucho". Esto es cierto para programas pequeños o quizás solamente para los diminutos. Sin embargo, para programas grandes y muy grandes, esta práctica ha ocasionado *costos* enormes a la hora de corregir defectos. El problema es que en los programas grandes normalmente participan muchos programadores cada uno haciendo una parte y las modificaciones no siempre las hace el autor de cada parte. Estas modificaciones no siempre son adecuadas, ya que el código, en proyectos medianos o grandes, tiene una estructura compleja que es difícil entender. Más aún, si el código no está comentado y escrito de una manera auto-explicativa, resulta casi imposible llevar a cabo modificaciones adecuadas aún para el mismo autor.

Para lograr la calidad del Software es necesario adaptar y adecuar los conceptos de control de calidad, aseguramiento de la calidad y mejoramiento continuo que han sido desarrollados para las otras industrias. Para lograr la calidad se debe ofrecer Software que proporcione, en el menor tiempo posible, buenos servicios, *costos* y *precios* bajos.

2.2.2 El software de calidad

El Software es útil en muchas áreas porque incrementa el *valor* del trabajo realizado por el usuario. Un Software bien hecho hace que el usuario logre más con menos, es decir, mejora su eficiencia y su eficacia. Cuando el usuario tiene a su servicio una herramienta de alta precisión y velocidad que además es fácil de usar, se dice que cuenta con un sistema de Software "bien hecho".

El Software es un producto que, a diferencia de otros, es muy versátil porque resulta muy fácil de modificar. Constantemente se están ofreciendo en el mercado nuevas versiones que brindan a los usuarios más y mejores servicios. Sólo es necesario editar el programa fuente y correr un compilador para tener una versión nueva del mismo. No es necesario, como en otras industrias, modificar procesos de producción o crear nuevas "piezas" para lograr un producto diferente. Esta versatilidad lo hace vulnerable ante errores humanos al momento de ser modificado. Cuando un programador se equivoca, se dice que ha introducido un "defecto". Un *defecto* es una inconsistencia del producto con sus *requerimientos*, y debe ser eliminado.

En este libro estudiaremos los defectos del Software y sobre todo las técnicas hasta ahora conocidas para lograr la creación de Software libre de defectos o al menos con la mínima cantidad posible.

El control de calidad del software es el conjunto de actividades destinadas a la detección y corrección temprana de defectos en el sistema de software que está en producción.

2.2.3 Definiciones de Calidad del Software

El concepto de Calidad del Software tiene varias interpretaciones y no se ha establecido un consenso. En lugar de plantear una definición, los libros de calidad enlistan las definiciones de los mas reconocidos en el área. A continuación mencionamos las mas relevanes. Por ejemplo:

- Robert Glass: el buen trabajo es aquel que maximiza s/(e*p), con s=cantidad de servicios otorgados, e=cantidad de errores, p=*precio*. Es decir, que el Software tenga más servicios con menos errores a menor precio.
- DeMarco: Se hace la pregunta: ¿Qué tanto nuestro Software cambia el mundo para bien?
- Phil Crosby 1979: indica que la calidad depende simplemente de la concordancia del Software con los *requerimientos* del usuario, tanto explícitos como implícitos.
- Watts Humphrey 1989: dice que se deben lograr niveles excelentes de aptitud para el uso del SW. Es decir, que el Software sea apto para usarse.
- IBM: "Market-driven Quality". Sólo el mercado dicta la calidad. Si se vende es bueno.
- ISO 9001: Es el grado en el que un conjunto de características cumplen con los requerimientos especificados.

2.2.4 Diferentes puntos de vista de la Calidad del SW

- Visión trascendental
- Visión desde el usuario
- Visión desde la manufactura
- Visión desde el producto
- Visión por valor

2.2.4.1 Visión trascendental

Desde este punto de vista, la calidad del Software es un ideal inalcanzable. Esto se basa en la idea de que el Software es siempre perfectible y también en que las necesidades del hombre están en continuo cambio y, como el Software no cambia por sí sólo para adaptarse a estos cambios, siempre es necesario estar modificándolo para acercarlo al ideal de cumplir con todas estas necesidades.

2.2.4.2 Visión desde el usuario

En esta interpretación, se dice que el Software es de calidad si es adecuado para los propósitos del usuario, es decir, si para el usuario es una buena herramienta y efectivamente le sirve para aumentar su productividad. Cabe hacer notar que lo que el usuario necesita no siempre es lo que se pidió desde el principio en la especificación de requerimientos. Otro factor que interviene aquí es el precio de venta del SW. Aunque no esté dicho específicamente cual es el *precio* máximo, si éste es excesivo para el cliente o usuario, entonces no es adecuado para sus propósitos y no es un Software de calidad.

2.2.4.3 Visión desde la manufactura

En este caso la calidad del Software se evalúa en función del grado en el que cumple con la especificación de requerimientos. Nótese que, aunque dicha especificación la hace el cliente o usuario, ésta puede no coincidir con lo que realmente necesita. Sin embargo, bajo el punto de vista de la manufactura, el Software es de calidad ya que se hizo lo que se acordó en la especificación. Entre los requerimientos puede estar incluido un límite máximo para el precio, lo que implica mantener los costos bajos.

2.2.4.4 Visión desde el producto

Según esta visión, el Software es de calidad si tiene buenas características, independientemente de lo que un cliente o usuario pida. Normalmente, el cliente no tiene un conocimiento profundo de las características del Software pero sin embargo las disfruta. Cuando el Software está hecho con características adicionales a lo que un cliente pide, éste adquiere mayor calidad. Les da más

a los clientes aunque no lo hayan pedido. Entonces la calidad está en función de las propiedades del Software independientemente incluso de su *precio*.

2.2.4.5 Visión por valor

La calidad del Software está dada por la cantidad de dinero que el cliente está dispuesto a pagar por él. Desde éste punto de vista, si el producto cumple con los *requerimientos*, y si tiene propiedades adicionales, el cliente tendrá una mayor disposición a pagar más por él.

2.2.5 Las *fallas* en el Software

2.2.5.1 Naturaleza de las fallas de Software

Una *falla* de Software es un comportamiento inadecuado que provoca resultados erróneos. Cuando por un *error humano* se introduce un *defecto* en el Software (en código o datos), este *defecto* produce una *falla*. Una falla de Software puede provocar un defecto adicional (modificando datos o código) el cual puede producir una nueva *falla*. De esta manera pueden producirse *fallas en cascada* como se ve en la Figura 2-1. Esto puede entonces repetirse muchas veces haciendo muy difícil la búsqueda del *defecto* que originó la primera *falla*.

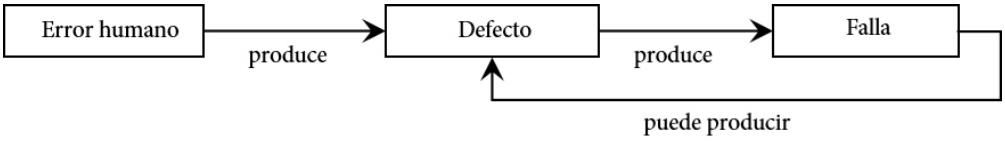


Figura 2-1 Relación entre error humano, defecto de Software y falla de SW

2.2.5.2 Importancia de las fallas de Software

Las *fallas* pueden ser desde inocuas hasta muy graves, dependiendo del uso que se le dé al SW. Por ejemplo, pueden provocar grandes pérdidas económicas, o incluso provocar la muerte de alguna o muchas personas.

Los sistemas computarizados en los bancos y en las casas de bolsa son ejemplos en donde una *falla* de Software puede generar pérdidas de dinero.

En los hospitales, en transporte aéreo, en sistemas de seguridad, en sistemas automatizados que manejan herramienta o material peligroso existe Software que, en caso de fallar, puede provocar tragedias que pueden costar muchas vidas humanas. Es por eso que en ese tipo de sistemas, es muy importante ofrecer una garantía de que el Software no tendrá fallas o por lo menos que, si se presentan, éstas no serán graves.

Por supuesto que en otros casos puede haber mucha mayor tolerancia a fallas. Todo depende del ambiente en el que opera el SW.

2.2.5.3 Origen y costo de las fallas

Las fallas tienen su origen en la premura por entregar un producto de Software y por lo tanto, en la poca dedicación a realizar suficientes pruebas. En otras palabras, no se asigna un presupuesto adecuado para darle calidad al Software creado.

Durante la fase de mantenimiento, en ocasiones es más fácil rediseñar el Software desde el principio que tratar de entender lo que ya está hecho para identificar lo que se hizo mal. Esto sucede principalmente cuando el código es complicado y no está escrito de forma clara. Incluso es común que cuando un programador no escribe un código claro, ¡ni el mismo puede entender el código que escribió!

Al reescribir programas que presentan fallas y que no son claros, se puede caer en la tentación de rediseñar de más, aún las partes que sí funcionaban. Esto trae como consecuencia la posibilidad de que se introduzcan nuevos defectos en donde no los había. Por lo tanto, hay que probar todo de nuevo lo cual incrementa el costo de desarrollo.

Se ha visto en la práctica que un 50% del costo de corrección de defectos encontrados en la fase de pruebas se deriva de errores humanos cometidos en fases de desarrollo mucho más tempranas. De haber sido detectados en esas fases, no sería necesario repetir una gran cantidad de pruebas ni hacer correcciones en todas las fases subsecuentes. Esto indica claramente que vale mucho la pena hacer un esfuerzo por corregir los errores cometidos lo antes posible[4].

2.2.6 Relación entre costo, precio y valor

La calidad del Software depende de la relación entre tres cantidades: costo, precio y valor. Estas cantidades indican dinero.

El *costo* es la cantidad de dinero necesaria para que sea posible crear el producto. El productor, que finalmente será el que lo venda, debe invertir una cantidad de dinero para poder ofrecer luego un producto terminado y venderlo.

4Pfleeger, Shari L. *Ingeniería de software. Teoría y práctica,1ª edición*. Editorial Pearson Education, Buenos Aires, 2002.

El *precio* es la cantidad de dinero que el productor del Software pide por la venta del producto.

El *valor* es la cantidad de dinero que el cliente (o el mercado en su totalidad) está dispuesto a pagar como máximo por el producto.

Para que pueda decirse que el Software tiene calidad, se debe cumplir la relación

$$C < P < V$$

Donde C es el costo, P es el precio y V es el *valor*. Obviamente, si el *costo* es menor al *precio*, el productor no pierde dinero. Si el precio es menor al valor, el cliente estará dispuesto a comprar el SW.

Esta relación se rompe cuando el productor incurre en costos demasiado elevados.
Cuando el costo excede al valor, la calidad se ha perdido por completo.

Otra manera de romper esta relación es cuando hay demasiada oferta de productos similares. En este caso el valor disminuye ya que la competencia por el mercado lleva al cliente a elegir la opción más barata.

La industria del Software tiene la particularidad de que la presencia de fallas en un sistema incrementa los *costos* muy rápidamente y en mucho mayor cantidad, al mismo tiempo que hace bajar el *valor* también muy rápidamente y en mucho mayor cantidad que en las demás industrias. Por esto, las fallas de Software tienen un impacto muy alto en la posibilidad de romper la relación costo-precio-valor. Cuando el Software falla se incurre en *costos* mucho más grandes porque se debe hacer mucho esfuerzo adicional relativo para corregirlas y, al mismo tiempo, el valor del Software disminuye también mucho porque el cliente pierde fácilmente la confianza y no querrá pagar por algo que está mal hecho. Lo anterior motiva la cancelación de contratos o a su no reanudación lo que ocasiona pérdida de dinero para del desarrollador.

2.3 Organización de la calidad: Actividades para el control y la garantía de la Calidad del SW

En la industria del Software, el control de calidad significa buscar defectos y corregirlos, al menor costo posible, antes de que produzcan fallas ante el cliente. En general, esto se logra al implantar la ejecución de una serie de pruebas específicamente diseñadas para verificar que el Software no falla, es decir, que cumple con los requisitos. Esta verificación se lleva a cabo durante todo el proceso de desarrollo y, por lo tanto, se verifican todos los productos intermedios generados. La idea es detectar los defectos en la etapa más temprana posible para evitar que éstos se propaguen hacia

las etapas siguientes lo que provocaría un incremento muy alto en los costos derivados de su corrección, ya que, al corregirlos, sería necesario repetir al menos parte del esfuerzo invertido en las etapas ejecutadas con el defecto incluido pero ahora sin éste. Cuanto antes se detecte un defecto menor será el costo de corregirlo.

Estos productos intermedios son los documentos que impactan en el producto final como la especificación de requerimientos, el diseño, módulos de código y también los documentos que indirectamente afectan a éstos como reglas de documentación, reglas de codificación, descripciones de procedimientos, listas de pruebas, etc.

Para verificar que un producto intermedio cumple con los requisitos para los que fue creado debe "pasar" ciertas pruebas. Estas pruebas no son siempre la ejecución en la computadora ya que los productos intermedios incluyen documentos que no necesariamente son programas ejecutables, aunque puede pensarse que estos son "ejecutados" por las personas involucradas en la siguiente etapa del desarrollo del Software y claro está que si contienen defectos, harán que estas personas produzcan documentos (o SW) con defectos también.

Entonces, para llevar a cabo el control de calidad, se usan diversas herramientas como revisiones entre colegas, inspecciones formales y pruebas. Se recomienda adoptar y desarrollar documentos de reglas generales que dicten los lineamientos que se deben seguir para lograr documentos de mayor calidad. Reglas específicas para cada fase del proyecto también son útiles. En el apéndice a se ilustra un ejemplo de reglas de codificación.

2.3.1 Revisiones entre colegas

Una de las técnicas para "probar" un documento o código es la *revisión entre colegas*[5]. El procedimiento para llevarlas a cabo es el siguiente y se ilustra en la Figura 2-2.

1. El documento es creado por una persona a la cual se le refiere como el autor. El autor se asegura de que el documento ya esté terminado y cumpla con los requisitos estipulados antes de entregarlo a su jefe inmediato superior (el líder).
2. El autor y el líder deben acordar una lista de personas que tengan la capacidad para revisar el documento. Mientras más expertos sean mejor. Dicha lista de revisores será incluida en el documento.
3. Manualmente o mediante un sistema automatizado, se hace llegar el documento a cada revisor y se le da un plazo razonable para que lo lea detenidamente y en base a sus conocimientos, experiencia y habilidades, encuentre defectos.
4. Si los revisores tienen algo que comentar acerca de la calidad del documento, deben hacerlo por escrito en una sección del documento especialmente reservada para esto. Cada comentario debe ir acompañado de la identidad del revisor.

5 Gilb T., Graham D., *Software Inspection*, Addison Wesley, 1993.

5. Al terminar el plazo para la revisión, el autor debe responder a cada comentario indicando si está o no de acuerdo con éste. Si no está de acuerdo, el revisor que hizo el comentario deberá contestar si está de acuerdo con la respuesta. Esto se repite mientras no se llegue a un acuerdo. Cuando se llega a un acuerdo, se dice que el comentario está cerrado.
6. Cuando los revisores piensan que no hay necesidad de hacer cambios al documento, deben indicarlo agregando un comentario diciendo que aceptan el documento.
7. El autor debe atender los comentarios en los que se haya acordado hacer modificaciones al documento y entregar una nueva versión corregida. En este caso se repiten los pasos 3 al 6.
8. Se declara el documento como "aceptado".

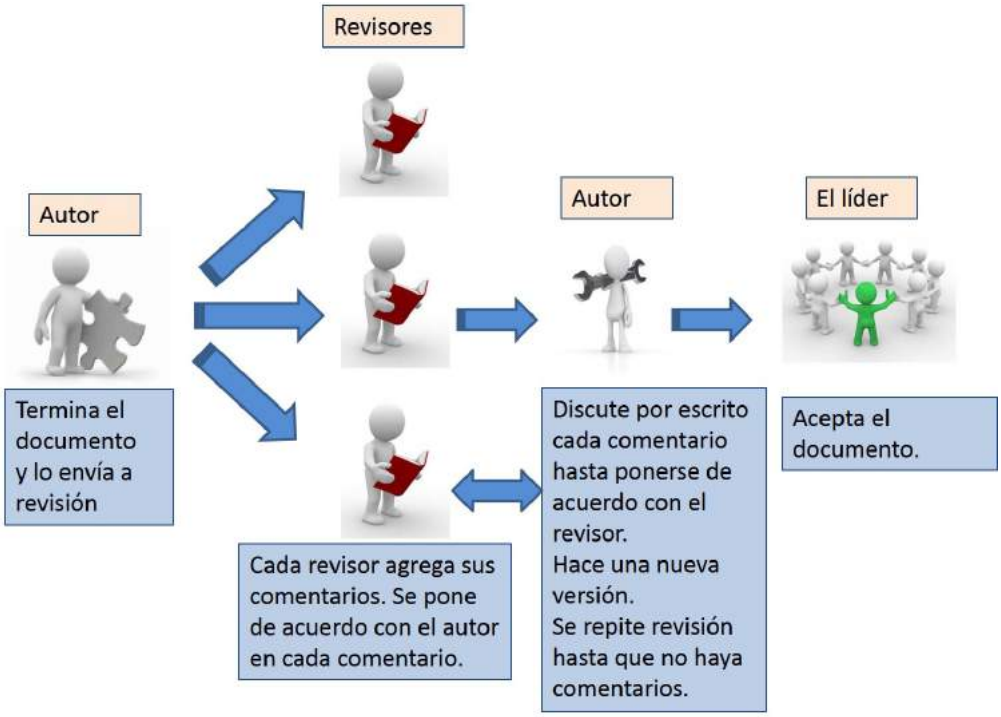


Figura 2-2: El proceso de la revisión entre colegas

2.3.2 Inspecciones

Otra técnica para revisar la calidad de documentos o código es *la inspección*. A diferencia de la revisión por colegas, la inspección se considera una técnica de revisión formal[6]. En la inspección no solamente se aprovecha la experiencia de los revisores sino que va más allá. Básicamente se revisa la consistencia entre dos documentos y se coteja que se cumplan las buenas prácticas. Estas buenas prácticas deben estar bien definidas y documentadas.

2.3.2.1 Terminología de inspecciones

Documento producto: es el documento que será sujeto a inspección. También se le llama simplemente *Producto*. Puede ser cualquiera de los documentos del proyecto o el código fuente, completos o partes de ellos.
Documento fuente: es el que contiene los requerimientos necesarios para poder producir el documento producto. Por ejemplo: el documento de especificación de requerimientos es fuente para producir el documento de diseño, el cual es a su vez un documento fuente del código.
Reglas y procedimientos: contienen información de la manera en que el documento producto debería ser escrito. Dependiendo de las decisiones que tome la empresa de desarrollo de SW, se pueden incluir por ejemplo: reglas de documentación, reglas de codificación, reglas de creación de reglas, etc. entre otros.
Listas de revisión o "checklists": son listas para ayudar a revisar que un documento cumpla con algún conjunto de reglas. Interpretan las reglas de un documento de reglas, en forma de preguntas.
Defecto: es una inconsistencia entre el producto y alguno de los siguientes: documento fuente, reglas, procedimientos, él mismo, el sentido común (aunque no esté por escrito).
Defecto mayor: es el defecto que al parecer provocará un mal funcionamiento.
Defecto mayor no-funcional: es el *defecto* que no provoca mal funcionamiento pero que sí causa costos adicionales o puede provocar la inserción de nuevos defectos a futuro.

6 Gilb T., Graham D., *Software Inspection*, Addison Wesley, 1993.

2.3.2.2 El procedimiento de inspección

Cuando se lleva a cabo una inspección las primeras veces, es muy común caer en algunos errores que demeritan su resultado. Por esto es bueno recordar lo que NO son y lo que SI son.

Las inspecciones NO son:

- × sesiones de optimización del diseño
- × para dar el visto bueno del grado de ambición del diseño
- × clubes de discusión
- × burocracia
- × guerras entre diseñadores

Las inspecciones SÍ son:

- ✓ revisión de consistencia
- ✓ cotejo de buenas prácticas
- ✓ ayuda cortés para el autor

El procedimiento de inspección se define mediante una serie de pasos a seguir bien definidos que son:

1. Planeación de la inspección
2. Verificación de criterios de entrada
3. Junta de inicio
4. Inspección individual
5. Junta de registro
6. Junta de lluvia de ideas
7. Edición
8. Seguimiento
9. Verificación de criterios de salida

A continuación se detallan estos pasos.

2.3.2.3 Planeación de la inspección

El líder del proyecto debe asignar el rol de "Líder de Inspecciones" a una persona que haya tomado un curso sobre *inspecciones* y sobre *líder de inspecciones* para que sea quien conduzca las actividades de inspección en la organización. Si no lo hay, como por ejemplo en un curso sobre inspecciones en el que se harán prácticas didácticas, se eligirá un líder de inspecciones pero éste será guiado por otra persona experta, como el profesor.

También se debe calcular el *costo* que debería llevar la ejecución de las tareas de inspección dentro del plan del proyecto y el plan de aseguramiento de la calidad del proyecto. Se debe asegurar que se disponga de los recursos necesarios tanto humanos como logísticos y hacer una calendarización de éstos para apoyar las actividades de inspección.

El líder de proyecto y el de inspecciones acordarán quienes serán las personas más adecuadas para llevar a cabo la inspección así como la lista de objetivos de cada inspección. El autor del documento producto debe ser siempre uno de los participantes. Debe darse prioridad a la eficacia que a la eficiencia, por ejemplo, puede asignarse un equipo de 4 a 5 personas en lugar de 1 o 2 lo cual hace más costosa la inspección pero la hace más eficaz. Se debe acordar con los participantes el lugar y la fecha para hacer la inspección. Todo esto debe quedar asentado por escrito en el plan.

El líder de inspecciones debe asignar a cada participante un rol dentro de la inspección diferente al de todos los demás. Esto puede lograrse si el documento producto se divide en partes y estas partes se reparten a diferentes participantes. También puede repartirse el conjunto de documentos fuente para que los participantes revisen consistencia con uno diferente cada uno. Las reglas asociadas al tipo de documento también pueden repartirse. Haciendo una mezcla de estas tres formas de repartir el trabajo pueden asignarse tareas distintas a cada participante. Lo más recomendable es traslapar la repartición de manera que haya redundancia pero no total. Por ejemplo: si son 4 participantes, el documento producto puede dividirse en dos partes más o menos iguales y asignar al primer participante la tarea de revisar la primera parte contra documentos fuente, al segundo participante la de revisar la segunda parte contra documentos fuente y las reglas aplicables, al tercer participante se le puede asignar la tarea de revisar todo el documento producto pero solamente contra documentos fuente y al cuarto la de revisar todo el documento pero contra reglas solamente. Así el mismo trabajo será hecho por más de una persona pero con diferente enfoque.

El líder de inspecciones debe revisar cuales documentos, además del documento producto, son necesarios para la inspección como: documentos fuente, reglas, listas de verificación, etc. Debe luego preparar estos documentos para proporcionárselos a los participantes de la inspección durante la junta de inicio.

Toda la información del plan de cada inspección debe ser asentado en un documento llamado el Plan Maestro de la inspección (uno por cada inspección).

2.3.2.4 Verificación de criterios de entrada

El líder de inspecciones debe verificar que el documento producto "califica" para la inspección considerando como mínimo los siguientes criterios de entrada (a la inspección):

- El autor confirma que el documento está terminado y que incluye todos los requisitos descritos en el o los documentos fuente del mismo y que cumple con las reglas asociadas a ese tipo de documentos.
- El documento producto ya fue revisado mediante las herramientas automatizadas disponibles como: procesadores de texto, compiladores, etc. para detectar errores como: ortografía, sintaxis, etc.
- El líder de inspecciones no puede hallar defectos en el producto a simple vista.
- Si alguno de estos criterios no se cumple, el líder de inspecciones debe posponer el inicio de la inspección hasta lograr que se cumplan.

2.3.2.5 Junta de inicio

El líder de inspecciones se reúne con los demás participantes y explica a cada uno el rol que tiene en la inspección que está por iniciar. Debe indicar claramente qué parte del documento fuente y contra qué documentos debe revisar la consistencia. Esto no debe impedir que, en caso de hallar alguna otra inconsistencia, los participantes la reporten y el líder de inspecciones debe dejar esto bien claro.

Se debe dejar claro también cual es la *tasa de chequeo* o velocidad de inspección adecuada para hacer la revisión. Se recomienda hacer la revisión con detenimiento, es decir, a tasas bajas como de 1 a 3 páginas por hora o, en caso de código fuente, de 20 a 50 instrucciones por hora.

El líder de inspecciones deberá instruir a los participantes acerca de la manera en que deben reportar las anomalías que vayan encontrando. Normalmente se usa un formato para cada participante en el que se registra la siguiente información para cada anomalía o *defecto* encontrado:

- Número consecutivo
- Lugar exacto en donde está el *defecto*
- Clasificación del *defecto*
- Regla que se rompe o requerimiento que no se cumple
- Descripción breve (sólo en caso de ser necesario)
- Número de veces que se encontró (no se pone el lugar exacto de cada una)

A las anomalías o defectos encontrados se les llama "*asuntos*" y deberán ser tratados posteriormente por el autor. Debe ser claro para los participantes lo que NO es una inspección para evitar que usen su tiempo de manera inadecuada.

La clasificación de los defectos se hace en función de su gravedad y hay tres niveles: Mayores, Mayores no-funcionales y Menores. Los Mayores son aquellos que clara y definitivamente provocan mal funcionamiento del producto. Los Mayores no-funcionales son aquellos que directa o indirectamente incrementan los *costos* del proyecto, es decir,que si no son resueltos el Software funciona bien pero que si son resueltos la empresa funcionaría mejor. Los menores son todos aquellos *defectos* que podrían ser corregidos o no sin causar daño a la empresa como errores al escribir algunas palabras pero que queda claro lo que se quería decir, entre otros muchos posibles casos.

2.3.2.6 Inspección individual

La meta es detectar defectos mayores tanto funcionales como no- funcionales. Se enfatiza que los defectos buscados son los mayores ya que son éstos los que realmente causan problemas y *costos* mucho mayores en el desarrollo del proyecto. Hay que ser suficientemente críticos como para vislumbrar posibles problemas en etapas posteriores en las que el documento inspeccionado es usado como documento fuente. Cualquier situación de este tipo debe reportarse como un asunto a tratar con el autor.

Se debe usar la velocidad de inspección recomendada en la junta de inicio. Esto permite conseguir una mayor efectividad para encontrar defectos. Es necesario cotejar cada detalle del documento producto con varios otros documentos, lo cual lleva su tiempo.

Es necesario lograr un alto grado de concentración en esta tarea. No debe hacerse durante más de 2 horas seguidas. Debe hacerse en un lugar apacible, sin ruido ni interrupciones. En caso de que solamente se estén encontrando defectos menores, debe revisarse si el grado de concentración logrado es el adecuado.

Durante la inspección individual, pueden detectarse problemas para encontrar los defectos mayores del documento y hallar solamente menores. Esto puede tener su origen en varias fuentes como pueden ser: que los documentos fuente o los documentos de reglas tienen deficiencias, no hay o son deficientes las listas de verificación, no hay buena concentración, falta de experiencia, incluso, podría darse el caso de que el documento fuente no tiene defectos mayores porque está basado en algún documento previo al que ya se inspeccionó y las adiciones que incluye el nuevo documento son pocas.

Además de registrar los defectos, es posible también registrar como un asunto cualquier pregunta que se quiera hacer al autor en caso de tener dudas cuya respuesta pueda derivar en haber hallado otro defecto. Esta pregunta se le hará al autor durante la junta de registro en el siguiente paso. Las preguntas deben clasificarse como tales en donde se clasifican los defectos.

Si durante la inspección individual surge alguna idea acerca de cómo mejorar el proceso de inspección o el de desarrollo de Software o el de manejo de proyectos, debe también registrarse como un asunto. Los asuntos de este tipo serán clasificados como sugerencias.

En general todos los documentos contienen *defectos* mayores. Cuando tiene muchos, es fácil encontrarlos, si tiene pocos se vuelve más difícil. Cuando no se pueden encontrar *defectos* mayores, debe informársele al líder de inspecciones para que él decida si se prosigue o detiene la inspección o si se toma alguna medida alternativa.

2.3.2.7 Junta de registro

El líder de inspecciones programa esta junta de acuerdo con los participantes. El propósito es reunir a todos para exponer cada asunto registrado por los participantes de manera breve de tal manera que tanto el autor (o autores en caso de hallarse *defectos* en otros documentos además del documento producto) como el líder de inspecciones queden enterados.

Durante esta junta uno de los participantes (o el líder de inspecciones) debe funcionar como escribano llenando una forma de registro de asuntos igual a la que usaron individualmente cada uno pero que enliste todos los asuntos para el autor. En esta hoja no deben repetirse los asuntos reportados por más de un participante, es decir, si el mismo *defecto* fue hallado por varios participantes, se registra una sola vez. Las preguntas al autor son respondidas por éste y, en caso de decidirse que sí hay un *defecto*, se registra un asunto más, clasificado como Nuevo. Si no se llega a un acuerdo rápidamente, la pregunta se registra como tal y se pasa al siguiente asunto sin más discusión y el autor decidirá cómo resuelve el problema. Los *defectos* Nuevos son un indicativo de que la junta fue provechosa por lo que el líder de inspecciones debe promover el que aparezcan éstos durante la junta. Las sugerencias son registradas también y serán atendidas luego por el líder de inspecciones.

El líder de inspecciones debe asegurarse que el autor haya entendido cada uno de los asuntos registrados.
Es importante que el líder de inspecciones conduzca la junta con mucho tacto cuidando que la forma en que el autor se entera de sus errores sea **la más cordial posible**. No debe permitirse ningún tipo de agresión al autor ni una actitud defensiva por parte del autor. El ambiente debe ser de respeto y debe sentirse que todos están allí para ayudar al autor a conseguir un buen documento.

Las discusiones técnicas deben también evitarse. Es responsabilidad únicamente del autor el corregir los defectos en el documento y lo hará a su manera. Las discusiones solamente llevan a perder el tiempo. Simplemente se le hace saber al autor que hay un asunto más por considerar.

Cada participante debe medir el tiempo que realmente usó durante la inspección y reportarlo en su hoja de registro. Esto permite hacer mejor planeación para próximas inspecciones. Se reporta también el número total de defectos mayores, defectos mayores no-funcionales, defectos menores, preguntas al autor, sugerencias.

Esta junta no debe durar más de 2 horas seguidas. El líder de inspecciones debe mantener un buen ritmo de registro para que no se alargue pero, de ser necesario, puede interrumpirla y continuarla en otra ocasión.

2.3.2.8 Junta de lluvia de ideas

Esta junta se hace justo después de terminar la anterior y no debe durar más de media hora. El objetivo principal es el de descubrir y analizar la causa de los defectos mayores encontrados.

Durante este análisis se espera que surjan sugerencias de cómo evitar la introducción de defectos en el futuro. El líder de inspecciones debe invitar a que los participantes propongan maneras de mejorar los detalles del procedimiento de inspección que usaron y en general cualquier parte del proceso de desarrollo que haya sido identificado como causa de la introducción de los defectos encontrados.

Todas las sugerencias serán registradas en un reporte especial para esta junta pero no serán discutidas. Si alguna sugerencia es "descabellada" será registrada también y el líder de inspecciones debe conseguir que no haya burlas ni agresiones. La idea es promover la participación activa de todos en el mejoramiento continuo de la organización. Las sugerencias serán tomadas por el líder de inspecciones para darles cause hacia quienes están dirigidas y verificará que haya alguna respuesta. Muchas de ellas pueden estar dirigidas a él mismo.

2.3.2.9 Edición

El autor del documento recibe la hoja de registro de asuntos y atiende a cada uno de ellos modificando respectivamente el documento. Es su responsabilidad el que el documento quede bien, así que deberá hacer un buen análisis de cada asunto antes de hacer cualquier modificación.

El autor deberá escribir un reporte de falla o T-record dirigido a aquellos documentos fuente o reglas en las que se hayan encontrado *defectos* durante la inspección (este documento se describe más adelante).

El autor debe reportar en la misma hoja de registro de asuntos cuánto tiempo le llevó hacer las modificaciones necesarias y deberá avisar al líder de inspecciones cuando haya terminado.

2.3.2..10 Seguimiento

El seguimiento consiste en que el líder de inspecciones revisa que cada uno de los asuntos entregados al autor haya sido atendido por éste. En caso de que alguno no haya quedado reflejado en el documento, el líder de inspecciones pedirá al autor una explicación y, si queda convencido, considerará la edición terminada. Si no, pedirá al autor que atienda todos los asuntos primero.

2.3.2.11 Criterios de salida

Para poder dar por terminada la inspección, el líder de inspecciones verificará que se cumplan los criterios de salida los cuales son:

- El seguimiento dio un resultado positivo.
- La velocidad de inspección fue la adecuada.
- La hoja de registro tiene ya toda la información incluyendo el esfuerzo invertido por el líder de inspecciones y el de edición por el autor.
- La densidad de defectos mayores remanentes en el documento es menor a cierto valor.

Una buena cifra para esta densidad es 0.25 defectos mayores por página. Esta cifra puede irse modificando en base a la experiencia.

Se considera que el número de defectos mayores remanentes depende de la efectividad de los participantes. Esta efectividad se supone alta cuando los participantes son expertos y baja cuando no. Una cifra posible para empezar sería un 40%, pudiendo ser elevada hasta un 60%. Por lo tanto, el número de defectos encontrados es, como máximo, el 60% de todos los defectos que el documento tiene. Este total se obtiene mediante una simple regla de tres: si el número de defectos encontrados es el 60%, entonces ¿cuántos defectos son el 100%?
Como sólo se encontraron algunos defectos, habrá una cantidad de defectos remanentes que no fueron encontrados y que siguen allí en el documento. Esta cantidad se puede calcular restando del total de defectos (el 100% del que se habló arriba) la cantidad de defectos encontrados.

Se considera que cuando se corrijan los defectos encontrados en el documento, el autor introducirá un defecto nuevo por cada 6 defectos que edite. La cantidad total de defectos nuevos introducidos será entonces un sexto de la cantidad de defectos encontrados.

Así, el total de defectos remanentes en el documento es la suma de los defectos no encontrados más los nuevos defectos.

Por último, se obtiene la densidad de defectos *remanentes* dividiendo la cantidad de defectos remanentes entre el número de páginas en el documento.

En caso de que alguno de los criterios no se cumpla, el líder de inspecciones puede decidir que es necesario iniciar de nuevo la inspección. Por ejemplo, si hay una densidad de defectos remanentes alta, sería una buena razón para reiniciar la inspección.

2.3.2.12 Algunas directrices importantes

- *Revisar el producto, no al productor*: se trata de quitar los *defectos* del producto sin atacar al autor. Hay que mantener siempre una actitud cordial, de ayuda.
- *Ser breve*: Para poder cumplir con el plan, será necesario cortar al que empiece a divagar en las juntas.
- *No intentar resolverlo todo*: se debe reportar al autor y dejar que éste resuelva a su manera.
- *Desarrollar listas de verificación*: Iniciar con una lista aunque sea muy corta y usarla así. Con la experiencia de varias inspecciones puede irse mejorando la efectividad de estas listas.
- Asegurar que todos tengan los recursos necesarios: Esto es responsabilidad del líder de inspecciones.
- *Instruir a los participantes*: Sobre el proceso y sobre todo del aspecto psicológico de éste.
- *Repasar inspecciones anteriores*: Aplicar cambios al procedimiento de inspección. Esto debe ayudar a conseguir mejores resultados.
- *Planear inspecciones sobre los productos principales*: Inspeccionar los documentos en la línea principal del desarrollo de Software produce los mejores resultados. Las reglas y listas de verificación pueden irse desarrollando conforme van siendo usadas.

2.3.2.13 Listas de verificación

Cuando no se cuenta con una lista de verificación, debe iniciarse con una lista mínima para luego irla desarrollando más. A continuación se muestra una lista de verificación que sirve para revisar un documento de diseño. Esta lista puede utilizarse como punto de partida para desarrollar la lista de cada empresa en particular.

- ¿Están reflejados los requisitos del Software en la arquitectura del SW?
- ¿se consiguió una modularidad efectiva?
- ¿se definieron las interfaces para los módulos y los elementos externos?
- ¿se diseñó una estructura de datos consistente con la información que se maneja?
- ¿en qué grado se consideraron los siguientes aspectos?
  - Facilidad de mantenimiento
  - Eficiencia
  - Preservación de la integridad de los datos
  - Facilidad de expansión
  - Reusabilidad
  - Independencia del Hardware o de otro SW
  - Manejo automático de errores
  - Facilidad de operación
  - Seguridad
  - Simplicidad

Un ejemplo de una lista de verificación para codificación es la siguiente:

- ¿está libre de errores menores?
- ¿se tradujo correctamente el diseño en el código?
- ¿tienen las funciones nombres adecuados?
- ¿es fácil seguir el diseño en el código?
- ¿se usaron las convenciones de formato adoptadas y aceptadas?
- ¿está libre de comentarios incorrectos, ambiguos o incompletos?
- ¿son apropiadas las declaraciones de tipos y de variables?
- ¿son correctas las constantes definidas?

Estas listas de verificación deben ser consistentes con los documentos de reglas de diseño y de codificación respectivamente. De hecho, estos documentos de reglas deben ser creados primero y a partir de ellos se deben crear las listas de verificación.

2.3.2.14 Posibles motivos para llevar a cabo inspecciones

El procedimiento de inspección tiene un costo relativamente alto comparado con la revisión entre colegas. Sin embargo tiene muchas ventajas que la hacen atractiva para quienes deciden si se llevarán a cabo o no.

Una motivación obvia es la de detectar defectos, corregirlos y evitarlos. Con esto se logra reducir costos ya que mientras más temprano sean corregidos los *defectos* se incurrirá en menos costos de repetición de tareas. También se da mayor grado de confianza en que los tiempos de entrega serán los esperados ya que la repetición de tareas retrasa la entrega.

Otras motivaciones pueden ser conseguir: mediciones de la calidad de los productos o de la calidad del proceso mismo de desarrollo, o mediciones del costo y valor del proceso de inspección para poder hacer planes más adecuados, o mediciones del estado de un proyecto en particular.

Estas mediciones pueden motivar el hacer más inspecciones para mejorar el proceso de desarrollo y al mismo tiempo el proceso de inspecciones.

Las inspecciones son buenas también como un medio para que los participantes que no sean expertos tengan un acercamiento a los productos elaborados por expertos y que así logren una capacitación más rápida. Además colaboran, en la medida posible, en el aumento de la calidad de los mismos.

Al inicio, durante las primeras inspecciones, no se logran resultados tan buenos como cuando ya se tiene experiencia. Cuando una persona ya ha participado en varios procesos de inspección, podrá dar mejores resultados que las que participan por primera vez en uno. Así que, cuando ya se cuenta con algunos expertos en inspecciones, es posible capacitar a más personas simplemente haciéndoles participar junto con los expertos para que sean éstos los que les enseñen.

Las inspecciones también sirven para motivar al personal ya que, al lograrse un producto de mayor calidad, se está haciendo más productivo a cada uno de ellos lo cual es satisfactorio.

Otra motivación para planear inspecciones pudiera ser la de ayudar al autor de un producto en caso de que no se haya previsto que el producto que se le encargó fuera tan complejo como resultó ser. Una inspección es la vía idónea para proporcionarle ayuda y lograr que resuelva los problemas en menor tiempo. Al disponer de varias cabezas analizando intensamente un producto se logra aumentar la efectividad en el proceso. Al mismo tiempo, como ya se dijo, el proceso completo resulta más eficiente ya que no se incurre en costos adicionales por retrasos.

2.4 Estándares y normas

Existen organismos que emiten estándares y recomendaciones que las empresas de desarrollo de SW pueden adoptar. En general es conveniente seguir dichos estándares pero no siempre es el caso, ya que puede ser que no apliquen en algunos casos por diversas razones. En particular, las empresas se verán más beneficiadas al adoptar estándares mientras más grandes sean y sus proyectos de desarrollo sean más complejos.

2.4.1 Estándares ISO

La International Standards Organization (ISO) emite todo tipo de estándares, entre ellos hay unos relacionados con la calidad. Por ejemplo:

- ISO 9000 Quality management systems – Fundamentals and vocabulary
- ISO 9001 Quality management systems – Requirements

2.4.2 Los Estándares de la IEEE

La IEEE emite estándares de diversos tipos. A continuación se enlistan algunos ejemplos de estos estándares.

- IEEE-STD-830-1998 *Recommended Practice for Software Requirements Specifications*, IEEE Computer Society: Estándar para especificación de requerimientos. La estructura de la Especificación de requerimientos de este Estándar se respeta en la mayoría de los documentos de Especificación en cualquier parte del mundo cuando se elaboran sistemas de software a nivel industrial.
- IEEE-STD-730-200*2 Standard for Software Quality Assurance Plans*, IEEE Computer Society: propone y determina el contenido de los planes de aseguramiento de la calidad.
- IEEE STD-829 *Standard for Software Test Documentation*, IEEE Computer Society, 16 December 1998. : Documentación recomendada para el proceso de pruebas del software.

2.4.3 Modelo de madurez de capacidades del SEI.

El Software Engineering Institute (SEI) define 5 niveles de madurez de capacidades de una organización dedicada al desarrollo de Software. Estos niveles se usan para calificar a las organizaciones ya sea por medio de un organismo externo o interno a ellas. A grandes rasgos, estos niveles son:

Inicial: herramientas aplicadas informalmente. La calidad depende del empeño y habilidad de los desarrolladores.

Si se rigoriza la administración y se asegura la calidad se pasa a *Repetible*.
Repetible: el proceso es estable y el control estadístico es repetible. Es decir, que ya se cuenta con una disciplina capaz de hacer mediciones en todos los proyectos.

Si se incorporan modelos, métodos y tecnologías de la ingeniería del SW (estándares) se pasa a *Definido.*

Definido: se logra un progreso continuo. Se cuenta con evidencia de que la calidad del proceso de desarrollo de SW va en aumento.

Si se hace una base de datos del proceso para comparar la calidad relativa de cada producto y proceso se pasa a *Administrado*.

Administrado: se tienen mejoras substanciales de calidad del SW creado y se pueden tomar las medidas adecuadas cuando se detectan problemas relacionados al proceso.

Si hay recopilación automática de datos del proceso y se usan éstos para mejorarlo de forma sistemática se pasa a *Optimizado*.

Optimizado: mayor calidad y cantidad. De aquí sólo queda seguir optimizando el proceso

2.5 Planes de Aseguramiento de la Calidad

2.5.1 El Aseguramiento de la Calidad del Software

El concepto de aseguramiento de la calidad se refiere a la prevención de la introducción de defectos en el Software y no a su detección como en el control de calidad. En este sentido, el aseguramiento de la calidad va más lejos y pretende hallar las causas de la introducción de los defectos.

Esto se logra mediante la ejecución de pruebas (incluyendo revisiones, inspecciones) en todas las etapas del proceso de desarrollo de Software. Se hace un análisis de los resultados de las pruebas para encontrar las causas de la introducción de defectos. Las causas encontradas sirven de retroalimentación oftware, ya que son la base para hacer cambios destinados a reducir el numero y la gravedad de los *defectos* que se introducen.

También se hacen auditorias sobre las actividades de desarrollo de Software y de administración del proyecto en busca de anomalías o diferencias con los procedimientos que se hayan adoptado. Las auditorías son otra forma de investigar porqué hay defectos y que tan correcta es la manera de hacer el Software. En base a los resultados de las auditorías, se toman medidas para mejorar.

El objetivo final es instalar un sistema de mejoramiento continuo del proceso de creación del SW. Si este sistema se hace bien, la causa de las fallas debe ser visible y pueden ejercerse acciones correctivas para

mejorar la calidad del proceso de creación de Software. Con lo que se mejora la calidad del proyecto actual y/o de los proyectos subsecuentes. En organizaciones grandes, todas las actividades relacionadas con el aseguramiento de la calidad las lleva a cabo un grupo de personas dedicadas a esto exclusivamente. En las pequeñas se asigna temporalmente a un grupo de aseguramiento de la calidad.

2.5.2 El estándar IEEE 730

El Plan de Aseguramiento de la Calidad del Software **(S.Q.A.P.** por sus siglas en inglés Software Quality Assurance Plan), es un documento que organiza el desarrollo del software con el fin de que el proceso de su creación siga pautas que aseguren la calidad del resultado.

Debe definirse la política de calidad de la empresa en general y la de cada proyecto en particular. El estándar IEEE 730 establece recomendaciones para crear planes de aseguramiento de la calidad. Según este estándar, las secciones que debe tener el plan son las siguientes:

2.5.2.1 Propósito.

Identifica el alcance y el propósito específico de la creación del plan para el proyecto en el que será aplicado. Debe enlistarse los nombres de los productos que serán cubiertos por el plan y su uso. Se especifica el porcentaje del proceso de desarrollo de Software que será cubierto.

2.5.2.2 Documentos de referencia.

Lista de documentos que son citados en el resto del plan incluyendo la versión y la fecha de cada uno. Hay que incluir por ejemplo los documentos en donde se solicita la creación de este plan como pueden ser las políticas de calidad de la empresa, legislación u otros.

2.5.2.3 Administración.

Se describe la organización para el aseguramiento de la calidad del proyecto, tareas, roles y responsabilidades. También debe incluirse una estimación de los recursos necesarios para el aseguramiento de la calidad.

2.5.2.4 Documentación.

Se identifican los documentos que rigen el desarrollo, verificación y validación, uso y mantenimiento del SW. Se enlistan cuales documentos deben ser revisados y con qué método y criterios de confirmación de calidad. Hacer referencia a la sección 6 de este plan en donde se detalla el calendario para estas revisiones.
La documentación mínima es:

- Descripción de requerimientos de Software

- Descripción de Diseño de Software
- Planes de Verificación y Validación
- Reportes de resultados de Verificación y de Validación
- Documentación de Usuario
- Plan de administración de configuración de Software

2.5.2.5 Estándares, prácticas, convenciones y métricas.

Se identifican los estándares que serán utilizados así como aquellas prácticas que se consideran de calidad, las convenciones adoptadas y las métricas que serán recolectadas durante el proceso de desarrollo del proyecto.

2.5.2.6 Revisiones de Software.

Se indica qué productos serán revisados, cuándo, con qué método y por quienes. Se enlista un mínimo de revisiones que deben programarse para cumplir el estándar.

2.5.2.7 Pruebas.

En este apartado se indica el plan de pruebas del proyecto. En caso de existir un plan de pruebas por separado, se debe citar en esta sección.

2.5.2.8 Reporte de problemas y acciones correctivas.

Se debe incluir aquí el método o procedimiento que describe la manera en la que se reportarán los problemas que se vayan encontrando y la manera en que estos reportes serán atendidos, incluyendo las responsabilidades asignadas.

2.5.2.9 Herramientas, técnicas y metodologías.

Se identifican las herramientas (Software o hardware), técnicas y metodologías que se usarán para llevar a cabo las tareas de Aseguramiento de la Calidad del Software. Para cada una se debe indicar para qué se usará, cuándo es aplicable o no y sus limitaciones.

2.5.2.10 Control de medios.

Aquí se determina el lugar exacto en el que serán guardados los productos del proyecto incluyendo copias de seguridad y respaldo. También se define cómo se protegerán dichos productos de accesos no autorizados o de daños.

2.5.2.11 Control de proveedores.

Se establece la forma en la que se podrá asegurar la calidad de los productos suministrados por proveedores para el proyecto. Además se debe establecer el método mediante el cual se asegura que tales proveedores reciben nuestros requerimientos de forma adecuada y completa. Para proveedores de software ya hecho, se debe indicar el método para asegurar que el software adquirido sea idóneo para el proyecto. Para proveedores de software por desarrollarse se debe mostrar que el proveedor cuenta con un Plan de Aseguramiento de la Calidad del Software según este estándar.

2.5.2.12 Recolección de registros, mantenimiento y retención.

Se indica qué documentos de Aseguramiento de la Calidad serán conservados y el método por el cual serán archivados, salvaguardados, conservados y durante cuánto tiempo.

2.5.2.13 Capacitación.

Se programan las actividades de capacitación necesarias para llevar a cabo las tareas de este Plan de Aseguramiento de la Calidad del Software.

2.5.2.14 Manejo de riesgos.

Se identifican los métodos para identificar, valorar, monitorear y controlar las áreas de riesgo que surjan durante el proyecto.

2.5.2.15 Glosario de términos

Se incluyen sólo aquellos términos exclusivos de este Plan de Aseguramiento de la Calidad del Software.

2.5.2.16 Procedimiento de modificación del Plan de Aseguramiento de la Calidad del Software e historial.

Se indica el procedimiento mediante el cual se harán modificaciones a este plan y mantener un historial de dichos cambios.

2.6 Resumen

Calidad del Software: El software de buena calidad debe contener la menor cantidad de defectos posible. La facilidad para modificar el software y ofrecer una nueva versión del producto, lo hace vulnerable a la introducción de defectos. Un defecto es una inconsistencia del producto con sus requerimientos.

El control de calidad del software : Es el conjunto de actividades destinadas a la detección y corrección temprana de *defectos* en el sistema de software que está en producción.

Técnicas de control de calidad: Los defectos en un sistema de software, se presentan no sólo en el código, sino también en los documentos (Especificación de Requerimientos, Diseño, Pruebas, etc.). Las técnicas más conocidas para lograr la creación de Software con la mínima cantidad posible de *defectos* son: las *revisiones entre colegas* y las *inspecciones*. Éstas reducen los costos de producción, ya que mientras más pronto se detecte un defecto, menor será el costo de corregirlo.

La técnica de *revisiones entre colegas* consta de los siguientes pasos.

- El autor del documento o código somete su producto a revisión una vez que lo haya terminado.
- Los miembros del equipo que están en la lista de distribución (revisores), revisan cuidadosamente el producto y aportan comentarios que ayuden a mejorarlo.
- El autor atiende los comentarios y genera una nueva versión.
- El proceso se repite hasta que el autor y los revisores estén de acuerdo.
- En función de los comentarios y la forma en la que éstos fueron atendidos, líder del proyecto acepta o rechaza la última versión.

La técnica de *inspecciones* se resume en el siguiente mapa conceptual.

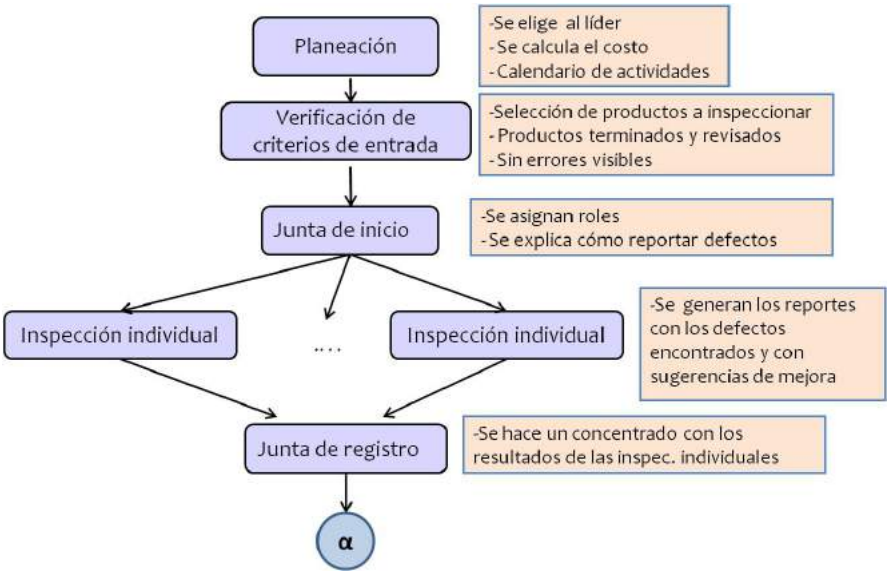


Figura 2-3: Las etapas de las inspecciones (I)

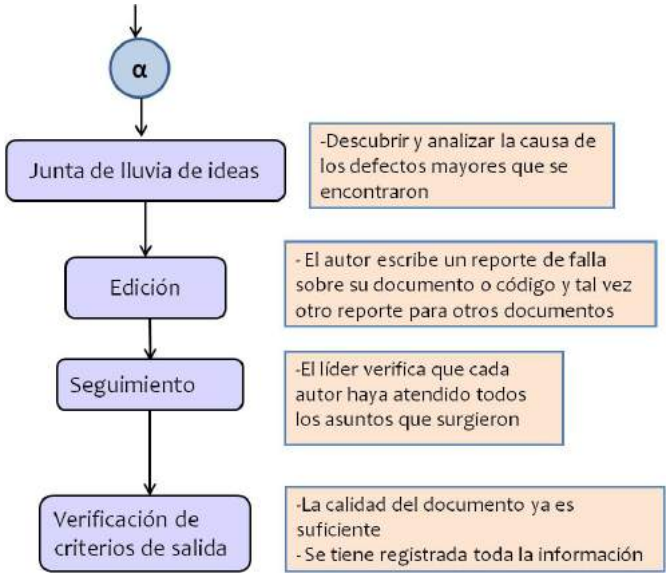


Figura 2-4: Las etapas de las inspecciones (II)

Aseguramiento de la Calidad del Software: Es un sistema de mejoramiento continuo del proceso de creación del SW. Tiene como objetivo encontrar las causas de la introducción de los *defectos* para poder prevenirlos. Para asegurar la calidad se ejecutan pruebas, revisiones e inspecciones en todas las etapas del proceso de desarrollo de un sistema y se analizan los resultados buscando las causas de los defectos. En base a estas causas, se hacen los cambios necesarios para reducir la introducción de defectos. También se llevan a cabo auditorías sobre las actividades de administración y desarrollo del proyecto. En base a los resultados de las auditorías, se ejecutan acciones destinadas a mejorar.

Plan de Aseguramiento de la Calidad: El Plan de Aseguramiento de la Calidad del Software (**S.Q.A.P.** por sus siglas en inglés Software Quality Assurance Plan), es un documento que planifica actividades adicionales al desarrollo del software mismo con el fin de asegurar la calidad del resultado.

2.7 Cuestionario

- ¿Qué significa *calidad en el software*?
- ¿En qué consiste el *control de calidad del software*?
- ¿Cuáles son las técnicas más conocidas para disminuir la cantidad de defectos del software y cuáles son los productos a los que se aplican?

- Explica cómo se lleva a cabo una *revisión entre colegas*.
- Describe el proceso de *inspección*.
- ¿Qué es el *Aseguramiento de la Calidad*?
- ¿Qué es el *Plan de Aseguramiento de la Calidad del Software*?
- ¿Cuál es el estándar que establece las recomendaciones mínimas para crear planes de aseguramiento de la calidad?

3. El Proceso de Pruebas del Software

3.1 Objetivos específicos del capítulo

- Conocer las actividades fundamentales de un proceso de desarrollo de software y su relación con los aspectos de calidad y pruebas.
- Comprender la importancia de la realización de pruebas en la Ingeniería de Software.
- Conocer el ámbito y alcance de un plan de pruebas y su proceso de ejecución.
- Conocer los roles y responsabilidades en un equipo de pruebas software.

3.2 El proceso de desarrollo de proyectos de software y la metodología de desarrollo

3.2.1 El desarrollo de proyectos SW y el ciclo de desarrollo de pruebas

Un proceso de desarrollo de software es el conjunto estructurado de las actividades requeridas para elaborar un sistema de software, estas actividades son: análisis y definición de requerimientos, diseño, implementación (codificación), pruebas y mantenimiento. A continuación se describe brevemente en qué consiste cada una de estas actividades[7].

Análisis y definición de requerimientos: Se trabaja con los clientes y los usuarios finales del sistema para determinar el dominio de aplicación y los servicios que debe proporcionar el sistema así como sus restricciones. Con esta información se produce el documento de "especificación de requerimientos del Sistema".

Diseño del sistema y del software: Durante el proceso de diseño del sistema se distinguen cuáles son los requerimientos de software y cuales los de hardware. Después se establece una arquitectura completa del sistema. Durante el diseño del software se identifican los subsistemas en los cuales estará compuesto el sistema y se describe cómo funciona cada uno y las relaciones entre éstos.

Implementación y pruebas: Consiste en codificar y probar los diferentes subsistemas por separado. La prueba de unidades implica verificar que cada una cumpla su especificación (proveniente del diseño).

7 Sommerville, Ian. *Ingeniería del Software,* Editorial Pearson, México, 2011.

Integración y validación del sistema: Una vez que se probó que funciona individualmente cada una de las unidades, éstas se integran para formar un sistema completo que debe cumplir con todos los requerimientos del software. Cuando las pruebas del sistema completo son exitosas, éste se entrega al cliente.

Mantenimiento: El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida y mejorar la implantación de las unidades del sistema para darle mayor robustez (y no nuevas funcionalidades).

3.2.1.1 Las pruebas de Software

Las pruebas de Software consisten en la ejecución de un sistema Software haciendo uso de la funcionalidad para la que fue diseñado, para verificar que dicha funcionalidad se ejecuta correctamente. El proceso se lleva a cabo siguiendo un protocolo de pruebas que consta de una lista de casos de prueba diseñados específicamente para este propósito.

Definición de Prueba: *Es el proceso de operar un sistema o componente bajo condiciones específicas, observar o registrar los resultados, y realizar una evaluación de algún aspecto del sistema o componente.*[8]

La fase de pruebas es indispensable en todo proyecto de Software. Ningún producto de Software con un mínimo grado de complejidad está exento de defectos que, al menos en potencia, generan *fallas*. Dado que en la actualidad se usa cada vez más el Software en aparatos en los que las consecuencias de las fallas son sumamente graves (muerte, fracaso en misiones importantes, quebrantos de seguridad, pérdidas financieras o sociales), no es viable entregar un producto sin haber pasado por un proceso de prueba que proporcione los niveles de confianza necesarios para poder usar ese SW.

3.2..1.2 El proceso de pruebas

El proceso de pruebas puede verse como parte del proceso de desarrollo de software, o como un proceso aparte. En la Figura 3-1 se ilustra el proceso de pruebas visto de manera independiente.

8 [IEEE 610.12-1990] de ANSI/IEEE, 1990

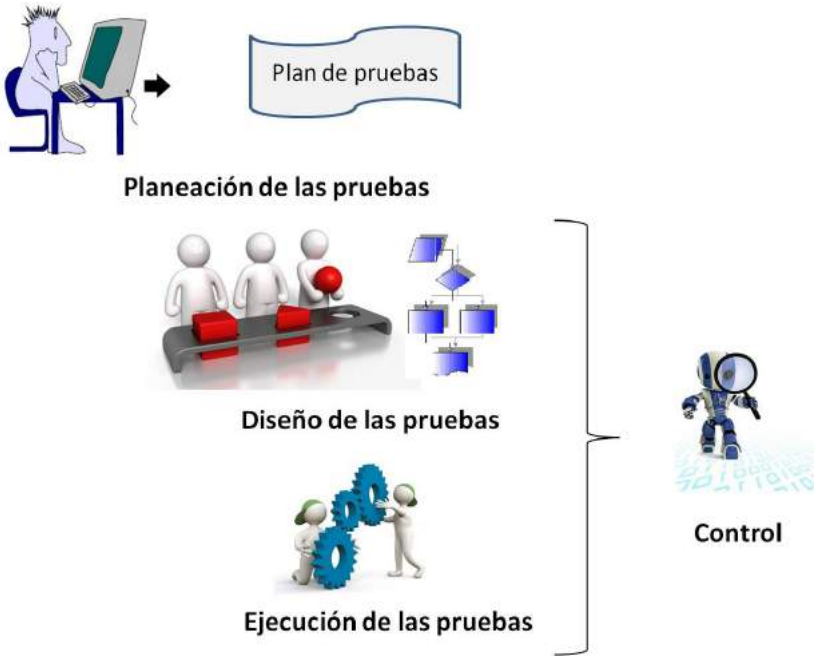


Figura 3-1: El proceso de pruebas

A continuación se describen las actividades de un proceso de pruebas ilustradas en la Figura 3-1.

Planificación: Durante esta fase se define el plan global de las pruebas: el plan de pruebas (agenda de las actividades de prueba, objetivos y requerimientos de las pruebas, personal que va a hacer las pruebas, recursos SW y HW), riesgos y definición de las herramientas a utilizar.

Diseño: Durante esta fase se identifican los elementos que deben ser probados. Se especifican las condiciones y datos de prueba en base a los elementos a probar. Se diseña la estructura de las pruebas y el ambiente que requieren. Las pruebas se estructuran, se organizan por grupos y categorías, y se les asignan prioridades en base a un análisis de riesgos.

Ejecución: Es el proceso de llevar a cabo las pruebas especificadas, observando y reportando los resultados. Algunos autores incluyen la instalación y configuración del ambiente para ejecutar las pruebas en la fase de diseño y otros en la de ejecución.

Control: Durante el diseño y ejecución de las pruebas se lleva a cabo un control que tiene como objetivos: monitorear el progreso, medir y analizar los resultados, realizar acciones correctivas y tomar decisiones estratégicas.

En la Figura 3-2 se muestra el proceso de ejecución de las pruebas. Se pueden ejecutar varias pruebas al mismo tiempo, cuando éstas son independientes entre sí.

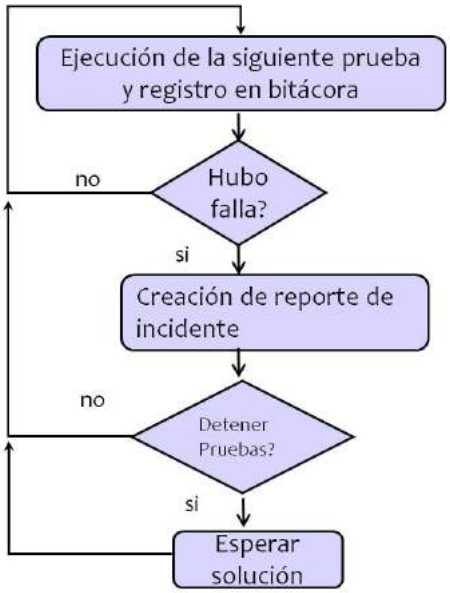


Figura 3-2: El proceso de ejecución de la pruebas.

Normalmente, un proyecto de Software utiliza un 40% del esfuerzo total de desarrollo en la fase de pruebas. Sin embargo, en ciertos proyectos se llega a invertir hasta un 80%.
¿Cómo se determina la cantidad de esfuerzo que será razonable invertir en la fase de pruebas? Si se trata de un Software que no es muy complicado y que fue diseñado en poco tiempo, ¿se debe invertir mucho tiempo en probarlo? Si se trata de un Software que implementa un juego para teléfonos celulares ¿cuánto tiempo hay que dedicar a probarlo? La respuesta a estas preguntas es un compromiso entre lo complejo del Software y lagravedad de las consecuencias. El primer caso puede ser el de un Software simple pero que será usado en un marcapasos en donde una falla puede ocasionarle la muerte al usuario. El segundo caso puede ser el de un Software que si falla no ocasiona graves daños pero puede tratarse de un Software muy complejo y difícil de crear. En conclusión, si el Software es más complicado que otro o si las consecuencias de una falla son más graves que la de otro entonces deberá usarse más tiempo de prueba. Comúnmente es la complejidad del Software lo que más se toma en cuenta al planear el esfuerzo dedicado a una fase de pruebas, y no se toma en cuenta la gravedad de las consecuencias de las fallas las cuales deberían tener un mayor peso.

En el estándar IEEE 1012 Software Verification and Validation se establece (entre muchas otras cosas) un sistema de 4 niveles llamados niveles de integridad del Software que determi-

nan la cantidad y el detalle de las actividades de Verificación y Validación del Software durante los diferentes procesos del ciclo de vida del SW. Los niveles de integridad son:

Nivel 4: El Software o componente del Software debe funcionar bien o de lo contrario habrá consecuencias graves (muerte, destrucción del sistema, pérdidas económicas o sociales) que no pueden ser mitigadas.
Nivel 3: El Software o componente del Software debe funcionar bien o de lo contrario el propósito para el que fue creado no se logrará causando serias consecuencias (lesiones permanentes, degradación del sistema, impacto económico o social) que pueden ser mitigadas por lo menos parcialmente.
Nivel 2: El Software o componente del Software debe funcionar bien o de lo contrario alguna de las funciones no será realizada causando consecuencias menores que pueden ser mitigadas totalmente.
Nivel 1: El Software o componente del Software debe funcionar bien o de lo contrario alguna de las funciones no será realizada causando consecuencias insignificantes que no requieren mitigación alguna.

3.2.2 Captura y Análisis de Requerimientos (Captación de requisitos y análisis funcional)

Los requerimientos especifican qué es lo que un sistema de software debe hacer (sus funciones) y sus propiedades esenciales y deseables. La captura de los requerimientos tiene como objetivo principal la comprensión de lo que los clientes y los usuarios esperan que haga el sistema.

La captura y el análisis de los requerimientos del sistema es una de las fases más importantes para que el proyecto tenga éxito. Como regla de modo empírico, el *costo* de reparar un error se incrementa en un factor de diez de una fase de desarrollo a la siguiente, por lo tanto la preparación de una especificación adecuada de *requerimientos* reduce los *costos* y el riesgo general asociado con el desarrollo.

El análisis de requerimientos es una tarea de ingeniería del software que permite especificar las características operacionales del software, indicar la interfaz del software con otros elementos del sistema y establecer las restricciones que debe cumplir el software. La tarea de análisis de los requerimientos es un proceso de descubrimiento y refinamiento, el cliente y el desarrollador tienen un papel activo en la ingeniería de requerimientos de software. El cliente intenta plantear un sistema que en muchas ocasiones es confuso para él, sin embargo, es necesario que describa los datos, que especifique las funciones y el comportamiento del sistema que desea. Por el otro lado, el desarrollador interroga, consulta y propone para resolver las necesidades del cliente.
El análisis de requerimientos proporciona una vía para que los clientes y lo desarrolladores lleguen a un acuerdo sobre lo que debe hacer el sistema, este acuerdo culmina con la elaboración del documento de *Especificación de Requerimientos*.

La *Especificación de Requerimientos* es un documento que define, de forma completa, precisa y verificable, los requisitos y el comportamiento u otras características, de un sistema o componente de un sistema. Este documento, es un producto que proporciona las pautas a seguir a los diseñadores del sistema.

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***.

Con la calidad de concordancia con requerimientos se pretende que el sistema implantado cumpla con los requerimientos que especificó el cliente. Si se tiene un buen proceso de diseño, se prevé buena calidad de implementación.

3.2.2.1 Las Solicitudes de Cambio

Cuando los Requerimientos de un nuevo proyecto son muy similares a los de uno o varios proyectos ya terminados, entonces se recurre al mecanismo de las *Solicitudes de Cambio* (Change Request, en inglés). La idea de las solicitudes de cambio es reutilizar documentos y código de proyectos anteriores para optimizar el desarrollo de proyectos nuevos. Una *Solicitud de Cambio* (SC) es un documento en el cual se indican los cambios que se deben hacer a un producto base, ya sea un documento o un módulo software.

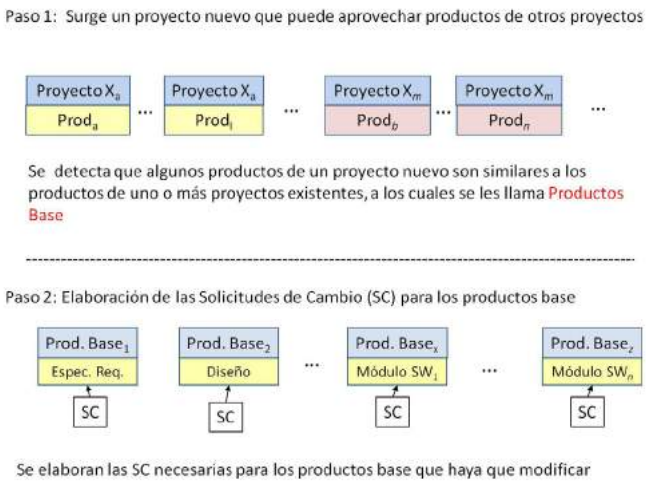


Figura 3-3 El proceso de las solicitudes de cambio (I)

Como se aprecia en la Figura 3-3, las Solicitudes de Cambio se elaboran cuando un proyecto nuevo aprovecha los productos de uno o varios proyectos existentes. Cada *Producto Base* pertenece a un proyecto ya existente, y se le asocia un documento de Solicitud de Cambio con la lista de modificaciones que se deben hacer para que este producto sirva al proyecto nuevo.

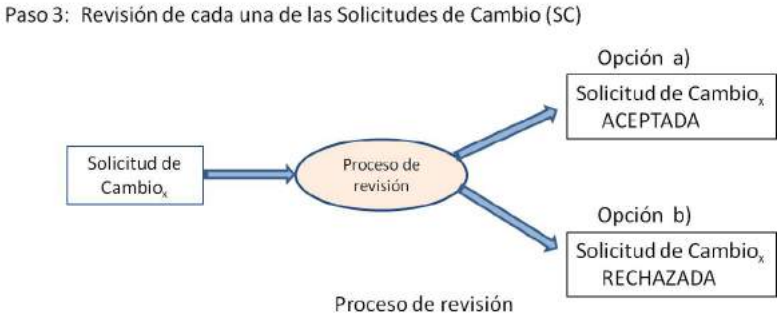


Figura 3-4 El proceso de las solicitudes de cambio (II)

A continuación, cada SC debe pasar por un proceso de revisión, como se ilustra en la Figura 3-4. Una vez que se termina la revisión de la SC, ésta adquiere el estatus de *Aceptada* o *Rechazada*, según sea el caso.

En el *paso 4 a)* la Figura 3-5 podemos apreciar que existen dos formas de utilizar una SC asociada a un documento base (Especificación, diseño, etc.) la primera opción (a1) es "aplicar la SC", esto significa generar un documento para el proyecto nuevo en el que se incluyen todos los cambios indicados en la SC. La otra opción (a2) es utilizar el documento base tal cual y, por separado, la SC que contiene la indicación de los cambios.

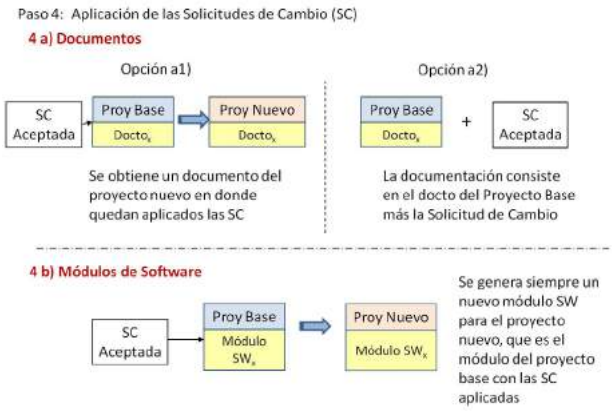


Figura 3-5 El proceso de las solicitudes de cambio (III)

En el *paso 4 b)* observamos que cuando hay una SC para un módulo Software, siempre será necesario generar el nuevo módulo del proyecto, el cual consiste en el producto base con los cambios de la SC aplicados.

Finalmente, en el *paso 5,* que se ilustra en la Figura 3-6, podemos observar que la configuración de un proyecto con solicitudes de cambio consiste en los documentos nuevos, a los cuales se les aplicó la SC conforme a la opción a1) del *paso 4 a)*, más los documentos de proyectos anteriores a los cuales no se les hizo ninguna modificación (si es que los hay). Adicionalmente, se tienen los módulos software nuevos, que consisten en los módulos SW base a los que se les aplicó su SC, más los módulos software de proyectos anteriores a los cuales no fue necesario modificar (si es que los hay).

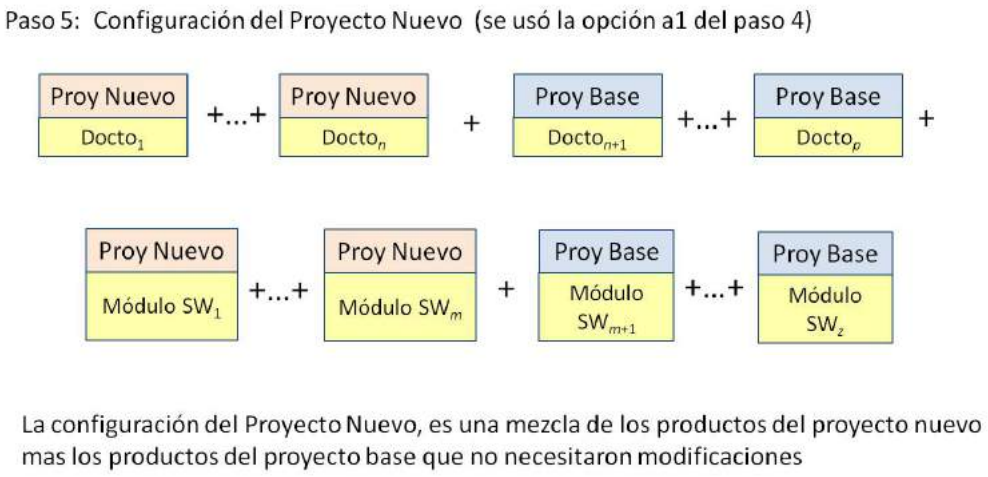


Figura 3-6: Configuración de un proyecto con Solicitudes de Cambio

3.2.3 Diseño e implementación del Software (Construcción del SW)

3.2.3.1 El Diseño

El diseño del Software combina la creatividad, intuición y experiencia del ingeniero de SW, con métodos de diseño, tomando en cuenta los criterios de calidad.

Los objetivos del diseño de Software son los siguientes:

1. Asegurar que se cumple con los requerimientos.
2. Anticipar y resolver problemas potenciales: Estos pueden ser la expiración de temporizadores, fallas en el suministro de energía, en las telecomunicaciones, en la base de datos, etc.
3. Distribuir el trabajo de implementación del sistema.

Durante el diseño se definen los componentes, las interfaces entre los componentes, las interfaces de usuario, la gestión de las tareas y la gestión de los datos del sistema que se va a implantar. El diseño debe satisfacer los *requerimientos* funcionales y también los atributos de calidad, tales como: modificabilidad, seguridad, desempeño y confiabilidad, usabilidad.

El diseño es un proceso iterativo al cual se le van haciendo mejoras, hasta obtener un diseño final. *La calidad del diseño se debe evaluar durante su elaboración, no después de terminarlo, y debe someterse a un proceso de revisión para minimizar los errores conceptuales.*

Con la calidad del diseño se busca que el sistema sea robusto y se tenga previsto como manejar los casos no exitosos.

Crear un buen diseño antes de comenzar a codificar mejora la robustez del producto, si hay cambios o modificaciones en la concepción del producto durante el proceso de desarrollo será mucho más conveniente atacar el problema desde el diseño que desde el código.

3.2.3.2 La implementación

La *implementación*, consiste en codificar lo que está indicado en el diseño. Durante la implementación se construyen la base de datos, las interfaces de usuario y todo el código que contiene la lógica del sistema.

Para codificar con calidad es necesario apegarse a las *Reglas de Codificación* establecidas por la empresa u organización responsable del sistema. De esta manera se promueve que el código sea legible. La *Lectura de Código* entre colegas ayuda a detectar a tiempo muchos errores que disminuyen el tiempo que se ocupará más tarde en probar y corregir.

Es importante saber usar el depurador de código "Debugger" para poder detectar los errores.

3.3 Roles en el proceso de las pruebas de software

3.3.1 El equipo de pruebas

Existen diferentes roles de las personas que participan en un equipo de pruebas. Éstos son el de líder, diseñador, probador (tester) o consultor.

El líder de prueba: es quien dirige el proyecto de prueba, es decir, escribe el plan de prueba, maneja los recursos (personal, SW, y HW), administra el presupuesto, coordina al equipo, supervisa el avance, detecta y resuelve problemas y toma decisiones estratégicas.
Los diseñadores de pruebas: hacen el diseño de los casos de prueba a partir de las especificaciones del producto. Es decir, analizan, revisan y evalúan los *requerimientos* del usuario, las especificaciones y los modelos para las pruebas, crean las especificaciones para las pruebas, instalan el ambiente de prueba, adquieren y/o elaboran los datos para las pruebas, revisan las pruebas desarrolladas por otros.
Los probadores (*Testers*): ejecutan los casos de prueba y reportan los resultados.
Un consultor: ayuda a dirigir al equipo y orienta sobre las mejores prácticas para llevar a cabo las pruebas.

3.4 Tipos de pruebas de SW

En las fases previas a la de pruebas, el ingeniero de Software es un creador y debe hacer su tarea con la confianza de que puede lograr el objetivo. Por el contrario, la fase de pruebas, se trata de hacer evidente lo que el Software no cumple, es decir, en cierta forma es una fase destructiva, es decir, durante las pruebas se busca detectar lo malo que hay en el SW. Cuando se diseñan las pruebas que deberán realizarse, se debe descartar la idea de que el Software está bien hecho. Por esto, si la misma persona que diseña el Software es la que diseña sus pruebas, puede haber un conflicto de intereses, pues por ejemplo, el diseñador podría diseñar casos de prueba a la medida, ya sea porque no puede ver más allá o porque intencionalmente prefiere que el Software pase las pruebas. De cualquier manera, en este caso las pruebas no tendrían una alta probabilidad de detectar todos los *defectos* y por lo tanto no serían unos buenos casos de prueba.

El proceso de pruebas debe ser visto como el proceso destructivo para encontrar errores (cuya presencia se asume) en un programa. Si el objetivo de la prueba es encontrar defectos, un caso de prueba exitoso es uno que hace fallar al programa.

Si se quisiera probar por completo un sistema, tendríamos que probar todas las posibilidades, es decir, hacer una prueba exhaustiva. La prueba exhaustiva requiere probar el comportamiento de un programa para todas las combinaciones validas e invalidas de entradas. Además se debe probar en cada punto donde se ingresan datos, y bajo cada estado posible del programa en ese punto. Un programa sencillo podría tener centenares o millones de combinaciones posibles, y el tiempo requerido para probarlas todas resulta prohibitivo. Entonces, un objetivo razonable es encontrar un conjunto de pruebas tales que se maximice el número de defectos encontrados pero con un número de pruebas finito y costeable.

Un buen caso de prueba es aquel en el que hay una buena probabilidad de hacer evidente, y con el menor esfuerzo posible, un defecto que no ha sido encontrado previamente.

Es importante aclarar aquí que el hecho de que el Software pase todas las pruebas no significa que no tenga defectos, sino que éstos no fueron detectados por las pruebas. Lo único que puede asegurarse es que cuando el Software no pasa las pruebas es que hay algún defecto ya sea en el Software o en el diseño de la prueba. Así que surgen las siguientes dos preguntas:

1. ¿Qué puede concluirse cuando el Software pasa un caso de prueba?

Respuesta: Que posiblemente el diseño del Software está bien hecho (pues podría no pasar en otros casos), o que posiblemente el caso de prueba está mal hecho, ya que quizás no se le exigió mucho al software.

2. ¿Qué puede concluirse cuando el Software no pasa un caso de prueba?

Respuesta: Hay dos posibilidades: 1) que el Software tiene un defecto y que el caso prueba tuvo éxito o 2) que el Software está bien pero el caso de prueba está mal diseñado. En cualquiera de estas dos posibilidades es necesario ejercer una acción correctiva ya sea en el Software o en el diseño del caso de prueba.

3.4.1 Técnicas de caja negra y de caja blanca

Existen varias técnicas para elaborar las pruebas. Una de las técnicas de pruebas son las de caja negra y de caja blanca. Como se observa en la **Figura 3-7**, las pruebas de caja negra son aquellas en las que el producto es visto como una caja sin luz en donde no es posible ver la forma en la que llevan a cabo sus funciones, lo único visible son sus interacciones con lo exterior, es decir, sus interfaces externas. Las pruebas de caja blanca son, por el contrario, aquellas en donde todos los detalles y la forma en la que funciona el producto están disponibles para el diseñador de pruebas, así que puede diseñar casos de prueba para cada uno de estos detalles.

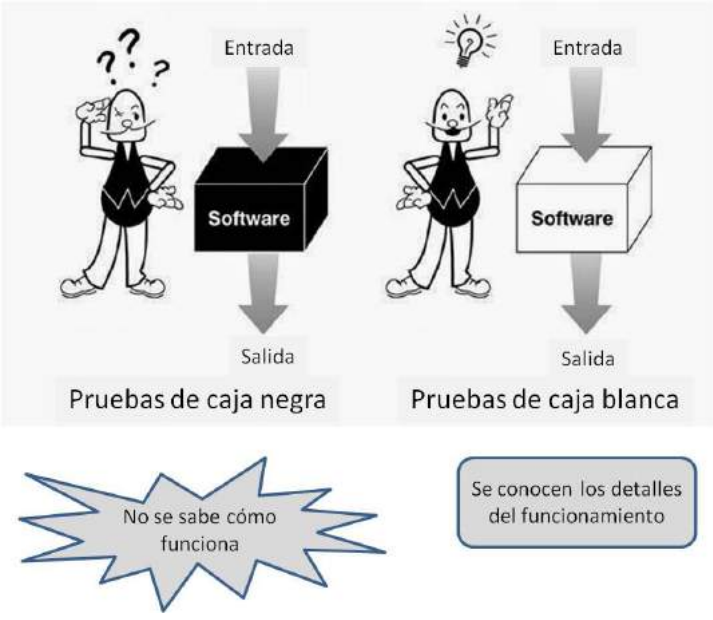


Figura 3-7: Pruebas de caja negra y caja blanca

El cliente no requiere ver la ejecución de pruebas de caja blanca ya que no le interesan los detalles sino el funcionamiento general de lo que se especificó durante el análisis de requerimientos. El cliente requiere sólo pruebas de caja negra a nivel general. Sin embargo para el proceso de desarrollo es necesario llevar a cabo pruebas con mucho más detalle ya sea de caja negra o de caja blanca.

Las pruebas de caja blanca tienen la ventaja de que pueden cubrir por completo toda la funcionalidad del Software exhaustivamente. Con ellas se puede probar toda la lógica interna de cada subrutina del código y todas las posibles rutas de flujo del programa. La desventaja es que para diseñarlas y ejecutarlas se requiere de mucho tiempo. Y podría ser imposible diseñar y ejecutarlas todas.

Las pruebas de caja negra son entonces una alternativa viable. En éstas, se diseñan casos de prueba en función de las interfaces diseñando las posibles entradas al sistema y las respuestas esperadas ante ellas. Estas pruebas se diseñan como "escenarios" en los que se señala la secuencia de las activaciones de las interfaces de un módulo a otro. Los accesos a archivos son también interfaces que deben probarse buscando casos donde se viole la integridad de los datos.

3.4.2 Tipos de Pruebas por su alcance

Las pruebas tanto de caja negra como de caja blanca se pueden diseñar a diferentes niveles. Dependiendo del nivel, las pruebas serán diseñadas en diferentes fases del desarrollo del SW. Se identifican 4 niveles: pruebas de módulo, de integración, de sistema y de aceptación.

3.4.2.1 Pruebas de módulo

También llamadas "*de componente*" o "*de unidad*" son pruebas que pueden ser de caja negra o de caja blanca y se diseñan para probar la funcionalidad de un sólo módulo. Para poder realizarlas a veces es necesario crear un módulo cuya única función es interactuar con el módulo a probar de manera que se cubran los casos de prueba diseñados para éste.

Estas pruebas deben diseñarse a partir del documento de diseño detallado si son de caja negra y a partir del código si son de caja blanca.

3.4.2.2 Pruebas de integración

Estas pruebas se realizan uniendo varios módulos para probar sus interfaces y las interacciones entre ellos. Se parte del supuesto de que es posible que haya inconsistencias en las interfaces, y/o que la interacción entre ellos puede generar comportamiento distinto al mostrado cuando se les prueba individualmente.

Estas pruebas son diseñadas normalmente como de caja negra a partir del documento de diseño de alto nivel y del documento de requerimientos.

3.4.2.3 Pruebas de sistema

El diseño de estas pruebas se basa principalmente en la especificación de requerimientos. Se ejecuta el sistema completo con el fin de verificar que la presencia de todos los componentes o módulos no afecta a la funcionalidad probada anteriormente en las pruebas de integración y en las de módulo. Estas pruebas se hacen para finalizar la fase de desarrollo del Software.

3.4.2.4 Pruebas de aceptación

Son similares a las pruebas de sistema pero orientadas a obtener la aceptación por parte del cliente. En estas pruebas se instala el sistema en un ambiente de operación real aunque posiblemente controlado y/o limitado. Y se diseñan exclusivamente a partir del documento de requerimientos. Normalmente son un subconjunto selecto de las pruebas de sistema.

3.4.2.5 Pruebas de Regresión

Las pruebas de regresión se llevan a cabo para verificar que no ocurrió una regresión en la calidad del producto luego de un cambio. Con estas pruebas se asegura que los cambios no introducen un comportamiento no deseado en otras áreas ya probadas. Implican la re- ejecución de alguna o todas las pruebas realizadas anteriormente.

3.5 Productos y herramientas de pruebas de software

El estándar IEEE Standard for Software Test Documentation 829-1998 describe el contenido de la documentación para la fase de pruebas. El conjunto de documentos de prueba que deberán ser creados dependerá de cada proyecto en función de la experiencia. Se recomienda usar estos documentos inicialmente sólo en pruebas de sistema y de aceptación para luego irlos introduciendo en pruebas de integración y de módulo progresivamente.

A continuación se describen brevemente los documentos de un proceso de pruebas.

3.5.1 Plan de pruebas

El *Plan de pruebas* es el documento en el que se describe el alcance, la estrategia, los recursos necesarios y la agenda o calendarización de todas las pruebas que se llevarán a cabo en un proyecto.

En éste se identifica qué partes y qué funciones serán probadas y cuáles no. Se describen las tareas que se deben realizar y quiénes son los responsables, por ejemplo: la administración del proyecto, el departamento de diseño de SW, el departamento de pruebas. También se define la participación que tendrá el cliente. Se identifican los riesgos existentes y se prevén las acciones necesarias en caso de una contingencia para poder cumplir con los tiempos de entrega.

El plan de pruebas contiene las siguientes secciones:

- *Información de control*: Identificador, autor, estado, lista de distribución, proyecto al que pertenece, etc.
- *Introducción*: Describe la relación de este plan con el plan del proyecto, con el plan de aseguramiento de la calidad, y con cualquier otro documento. Se indica qué es lo que se va a probar en términos generales.
- *Partes que se prueban y las que no se prueban*: Debe incluir la lista de los módulos que serán probados y los que no.
- *Funcionalidad que se prueba y la que no*: Se enlistan los documentos de diseño de prueba incluidos en el plan y una breve descripción. La funcionalidad que no se prueba debe también indicarse.
- *Estrategia general de las pruebas*: Indicar qué tipo de pruebas se harán y quienes o qué grupos las harán. Estimar el tiempo que deberá llevarse toda la fase de pruebas. Describir cómo se medirá el alcance de las pruebas (porcentaje del código que será ejecutado) y el grado de avance en ellas.
- *Criterios generales de aprobación*: describir los criterios para determinar si un módulo pasó o no la fase de pruebas.
- *Criterios de suspensión y de reanudación*: Determinar en qué casos debe suspenderse el proceso y bajo qué circunstancias puede reanudarse. Puede usarse por ejemplo un criterio en el que si el Software no pasa un cierto número de pruebas, se debe suspender las pruebas hasta que se tenga una nueva versión del SW. Otra posibilidad para detener las pruebas es que el Software no pase una prueba crítica o que se exceda un cierto tiempo prefijado.
- *Documentación de pruebas que debe entregarse*: lista de documentos generados antes y durante las pruebas. Debe incluir al menos toda la documentación descrita en el estándar IEEE 829.
- *Tareas*: preparación, ejecución, monitoreo, decisiones.
- *Requisitos para iniciar*: HW, SW, qué herramientas, qué documentos.
- *Responsabilidades*: cada tarea debe asociarse con un responsable.
- *Capacitación*: El nivel de capacitación mínimo que los responsables de las tareas deben tener para llevarlas a cabo.
- *Calendarización*: En lo posible determinar fechas para cada actividad y tarea.
- *Riesgos y planes de contingencia*: prevenir cualquier contratiempo y cómo resolverlo. Pueden considerarse inasistencias del personal, *fallas* externas (de energía,

huelgas), etcétera. Hay que considerar trabajar horas extras u otras soluciones adecuadas.

3.5.2 Diseño de pruebas

El *Diseño de pruebas* es un documento en el que se describe un grupo de casos de pruebas relacionadas entre sí. Cada documento de este tipo debe estar referenciado en el plan de pruebas y contiene las siguientes secciones:

- *Información de control*: Identificador, autor, estado, lista de distribución, plan de pruebas al que pertenece, etc.
- *Descripción*: una descripción general de la funcionalidad que se prueba y su relación con los documentos de *requerimientos* y/o de diseño.
- *Procedimiento de prueba*: describir cómo se deben llevar a cabo en general las pruebas abarcadas por este diseño. Incluir si deben usarse herramientas adicionales como comparadores de texto u otras.
- *Lista de casos de prueba*: Listar en breve los identificadores y los nombres de cada caso de prueba incluido en el diseño de prueba. Los casos de prueba pueden ser de caja negra o blanca.
- *Criterios de aprobación*: Describir cómo determinar si el conjunto de pruebas puede considerarse como aprobado.

3.5.3 Casos de prueba

Un *Caso de prueba* es un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y pos-condiciones de ejecución, desarrollados con un objetivo particular o condición de prueba.

En cada documento de este tipo se describe un sólo caso de prueba el cual debe ser parte de un diseño de prueba. Este tipo de documento contiene las siguientes secciones:

- *Información de control*: Identificador, autor, estado, lista de distribución, diseño de prueba al que pertenece, etc.
- *Descripción*: Parte y caso que se prueba. Establece la relación con otros documentos como el diseño de prueba, el plan de prueba, el de diseño y el de *requerimientos*.
- *Datos de entrada*: los parámetros exactos que se requieren para iniciarla o los datos de entrada que se reciben por ejemplo de un archivo o las respuestas recibidas desde otros subsistemas. Las condiciones iniciales, es decir, el estado de cada variable que afecta a la prueba. Puede incluirse el estado de la pantalla como una condición inicial.
- *Datos de salida esperados*: resultados entregados (valores) y/o los llamados a otros subsistemas y sus parámetros y/o el estado final de las variables o de los archivos. Puede incluirse el estado de la pantalla como estado final.

- *Requisitos de configuración*: describir qué módulos deben estar y qué hardware.
- *Procedimientos o herramientas*: Describir lo que debe hacerse y con qué herramientas. De ser necesario puede hacerse referencia a otros documentos en donde se describe el procedimiento.
- *Dependencia o relación con otros casos de prueba*: decir si hay algún otro caso que deba ejecutarse antes de este o después. Decir si algún otro caso depende del resultado de éste.

3.5.4 Procedimiento de prueba

Un *Procedimiento de prueba* describe los pasos a seguir para evaluar cierta funcionalidad de un sistema o subsistema. Contiene las instrucciones detalladas para la configuración, ejecución y evaluación de los resultados de uno o varios casos de prueba. Un procedimiento de prueba contiene las siguientes secciones.

- *Información de control*: Identificador, autor, estado, lista de distribución, diseño de prueba al que pertenece, etc.
- *Propósito*: descripción breve y referencia a los casos de prueba y diseño de prueba relacionados.
- *Requisitos especiales*: ambiente necesario, habilidades o capacitación necesaria, procedimientos previos a la ejecución de este procedimiento.
- *Pasos*: describir los siguientes pasos si aplican:

  Registro: modo en el que deben registrarse los eventos de la prueba como: resultados, incidentes, etc.

  Instalación: la secuencia de acciones para preparar la ejecución de la prueba.

  Inicio: secuencia para dar inicio a la prueba.

  Prosecución: acciones necesarias durante la ejecución.

  Medición: descripción de la manera en que se harán mediciones.

  Suspensión: acciones necesarias para suspender la prueba.

  Reinicio: identificar puntos en donde se puede reiniciar y las acciones necesarias.

  Detención: acciones para detener en orden la prueba.

  Restauración: acciones para restaurar el ambiente para la prueba.

  Contingencias: acciones en caso de haber contingencias.

En el apéndice B se ilustran dos ejemplos de procedimeintos de prueba.

3.5.5 Bitácora de pruebas

En este documento se registran brevemente todos los eventos en orden cronológico que suceden durante las pruebas. Contiene las siguientes secciones:

- *Información de control*: Identificador, autor.

- *Descripción*: cualquier información que aplique a todos los eventos registrados. Lista de módulos que se están probando. Características del ambiente que se está usando como: lugar, equipo, SW, etc.
- *Eventos*: registrar fecha y hora del evento y una descripción breve. El tipo de eventos que pueden registrarse incluyen:
  Inicio de procedimiento. Incluir identificador del procedimiento y caso de prueba, personal que está ejecutándolo y la función de cada uno.
  Resultado de la prueba. Incluir todo lo observado como mensajes o comportamientos del SW.
  Cambios en el ambiente de prueba. Substitución de HW, etc.
  Eventos anormales. Describir el evento y las condiciones previas y posteriores.
  Generación de reportes de incidente. Registrar el identificador del reporte de incidente cada vez que se genere uno.

La Figura 3-8 muestra la relación entre los documentos de un proceso de pruebas. Los Diseños de Prueba, los Procedimientos de Prueba y los Test Log están asociados a un Plan de Pruebas. Cada Diseño de Prueba contiene un grupo de Casos de Prueba. Cada Caso de Prueba se asocia a un Diseño de Prueba, pero también puede asociarse a uno o a varios Procedimientos de Prueba.

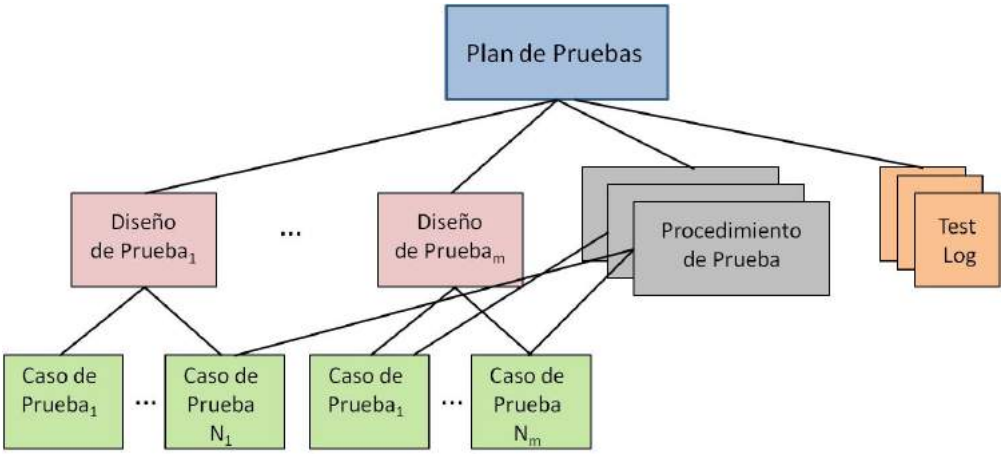


Figura 3-8 Los documentos asociados al Plan de Pruebas y las relaciones entre éstos.

3.5.6 Reporte de transmisión o entrega de módulo

Este documento identifica los módulos Software que han sido entregados al departamento de prueba y que, por lo tanto, se pueden iniciar o continuar las pruebas sobre ellos. Sirve como un indicador de que el departamento de diseño ya terminó su labor y para hacerle saber al departamento de pruebas que hay módulos que requieren ser probados.

Sus secciones son las siguientes:

- *Información de control*: Identificador, autor.
- *Módulos*: lista de módulos transmitidos y de los responsables de cada uno.
- *Ubicación*: lugar en donde el personal de pruebas puede ubicar y tomar los módulos entregados.
- *Estado*: Indicar para cada módulo si se entregó completo o no de acuerdo con la documentación. Indicar, en caso de ya haber sido probado, la lista de reportes de incidente o T-records que fueron atendidos en esta entrega. Indicar si los cambios afectan a la documentación del módulo (por ejemplo dentro del documento de diseño) y si los cambios en esa documentación ya fueron realizados.
- *Aprobación*: nombres y firmas de las personas que deben aprobar esta entrega.

3.5.7 Bitácora de pruebas (Test Log)

Registra los eventos durante la prueba. Cada vez que se ejecuta una prueba se registra en este documento. Contiene la siguiente información:

- Identificador, autor
- Descripción
- ¿A qué plan de pruebas pertenece?
- Lista de eventos (en breve) indicando hora
- Inicio/fin de tareas, observaciones, condiciones existentes, eventos anormales.
- Identidad del reporte de incidente correspondiente (si existe)

3.5.8 Reporte de incidente de prueba o T-record (Trouble-record)

En un reporte de incidente se registra un evento, ocurrido durante la ejecución de alguna prueba, que requiera investigarse. Normalmente se trata de reportes de fallas en el resultado de una prueba lo que indica la posibilidad de que existan defectos en el SW. Otro tipo de eventos puede también necesitar investigación aunque no sea por parte del departamento de diseño. Es posible que una falla tenga su solución en el mismo departamento de pruebas pero que el personal que lo reporta no sabe o no puede en ese momento resolverlo.

Un reporte de error debe contener: información de control, resumen, descripción e impacto. A continuación se explica en qué consisten cada uno de estos aspectos.

- *Información de control*: Identificador, autor.
- *Resumen*: descripción del incidente, lista de módulos involucrados, identificador del procedimiento de prueba, del caso de prueba y de la bitácora de pruebas.

- *Descripción*: detalles del incidente. Debe incluir:
  Entradas
  Salidas esperadas
  Salidas obtenidas
  Anomalías
  Fecha y hora
  Paso del procedimiento en el que se dio
  Ambiente
  Repeticiones
  Probadores
  Observadores
- *Impacto*: hasta donde el probador sepa debe indicar si este incidente afecta al plan de pruebas o al documento de diseño de prueba o al de especificación de diseño del SW, etc.

En la Figura 3-9, donde se ilustra el primer paso del proceso de los Reportes de Error, se puede apreciar que éstos son el resultado de la detección de un defecto en un producto de software. Se debe elaborar un Reporte de Error por cada defecto encontrado.

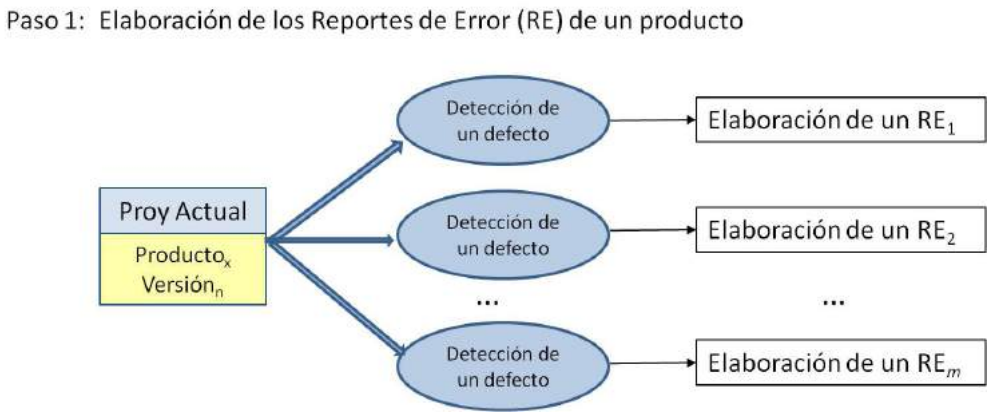


Figura 3-9: Ciclo de los Reportes de Error I

Cada Reporte de Error (RE) debe pasar por el proceso de revisión. Al finalizar este proceso, el RE tiene un estatus final de "*Aceptado*" o "*Rechazado*", como se ilustra en el paso 2 de la Figura 3-10.

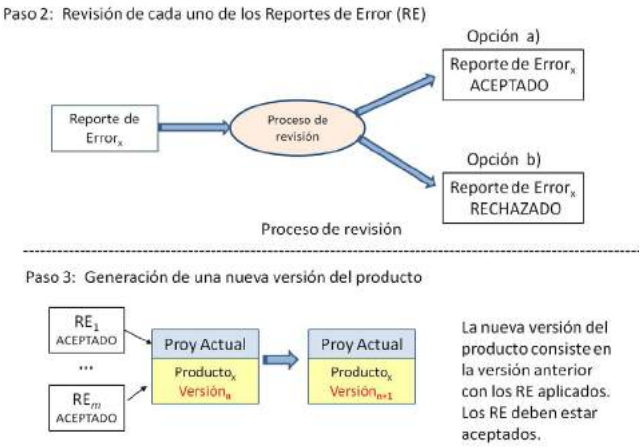


Figura 3-10: Ciclo de los Reportes de Error II

Finalmente, se debe generar una nueva versión del producto, la cual debe incluir la atención de cada uno de los reportes de error. Como se aprecia en el paso 3 de la Figura 3-10, la nueva versión que se genera tiene el número consecutivo a la versión anterior, por ejemplo, si el producto es el documento de diseño, versión 2, entonces la nueva versión será la 3.

3.5.9 Resumen de pruebas (Test Data Summary)

El resumen de pruebas sirve para resumir los resultados de todas las pruebas y poder evaluarlos. Éste debe contener:

- *Información de control*: Identificador, autor.
- *Resumen*: indicar qué módulos y qué pruebas se hicieron y en qué ambiente así como una lista de documentos previos a las pruebas y documentos de pruebas relacionados con cada módulo.
- *Variaciones*: explicar cuáles fueron las variaciones que se produjeron a raiz del proceso de pruebas en los módulos y en cualquier documento relacionado incluyendo a los documentos de pruebas.
- *Cumplimiento de criterios de aprobación*: establecer en qué grado se cumplieron los criterios de aprobación descritos en el plan de pruebas.
- *Resumen de resultados*: la cantidad total de pruebas que pasaron y la cantidad de reportes de incidente generados y cuántos de éstos derivaron en la corrección de algún *defecto* del Software y cuantos fueron resueltos de otra manera. Listar todos los incidentes que no fueron resueltos.
- *Evaluación*: se hace la evaluación de cada módulo explicando qué funcionalidad se probó y se basa en los resultados de las pruebas relacionadas con cada uno. Se hace una estimación del riesgo de *falla* para cada módulo.

- *Resumen de actividades*: describir a grandes rasgos las actividades realizadas y los eventos que surgieron. Indicar cuantos recursos fueron utilizados tanto humanos como logísticos y el tiempo total que duró la fase de pruebas.
- *Aprobación*: Fecha, firmas y nombres de cada una de las personas que deben dar su aprobación de éste reporte.

3.5.10 Productos comerciales y de código abierto

Existen herramientas comerciales y de código abierto que apoyan diversos aspectos de proceso de pruebas. Estas herramientas se pueden clasificar como sigue.

- Herramientas para la administración de las pruebas y su proceso: Ayudan en el seguimiento de incidentes, la gestión de la configuración y la administración de *requerimientos*.
- Herramientas para apoyar el proceso de revisión y de modelado.
- Herramientas para el diseño de las pruebas y para la preparación de datos de prueba.
- Herramientas de ejecución de casos de prueba, de pruebas unitarias, comparadores.
- Herramientas de desempeño, de carga y de estrés, y de monitorización.

Utilizar herramientas de pruebas tiene ventajas y desventajas. Entre las ventajas 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 las desventajas de las herramientas de pruebas hay que mencionar que se requiere esfuerzo adicional para aprender a utilizarlas. Además, se podría 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, o podría generarse demasiada confianza en una herramienta y sustituirla todas las pruebas manuales, lo que podría disminuir el hallazgo de los defectos.

3.6 Resumen

En el mapa conceptual de la Figura 3-11 se resumen las actividades principales de un proceso de desarrollo de software.

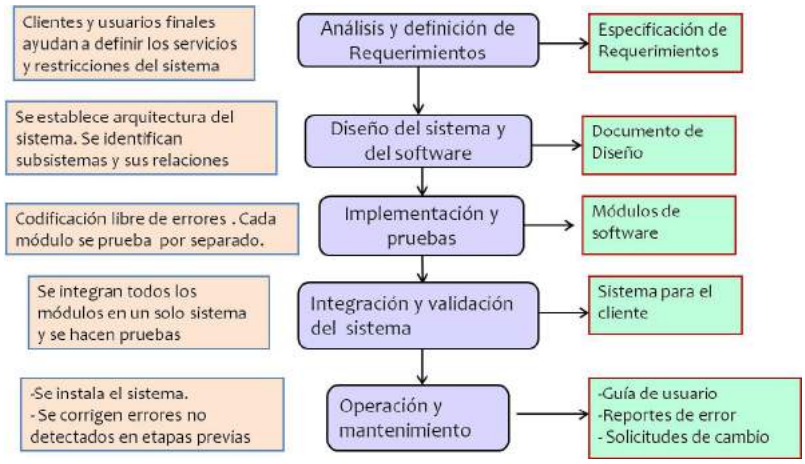


Figura 3-11: Las actividades básicas en un proceso de desarrollo de software

Pruebas del software Consisten en la ejecución de un sistema Software haciendo uso de la funcionalidad para la que fue diseñado, para verificar que dicha funcionalidad se ejecuta correctamente. El proceso se lleva a cabo siguiendo un protocolo de pruebas que consta de una lista de casos de prueba diseñados específicamente para este propósito.

En el mapa conceptual de la Figura 3-11 se resume el proceso de pruebas

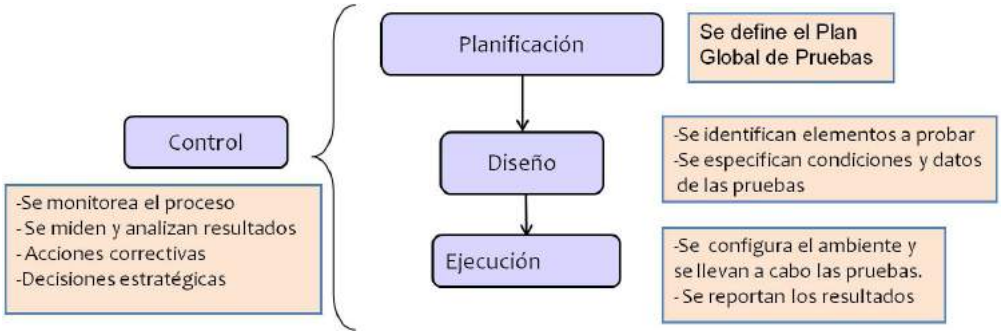


Figura 3-12: El proceso de pruebas

Pruebas de aceptación son las que permiten validar que el producto satisface los *requerimientos*.

Solicitudes de Cambio se elaboran cuando un proyecto nuevo aprovecha los productos de uno o varios proyectos existentes. Cada *Producto Base* pertenece a un proyecto ya existente, y se

le asocia un documento de Solicitud de Cambio con la lista de modificaciones que se deben hacer para que este producto sirva al proyecto nuevo.

Calidad en el Diseño el diseño es un proceso iterativo al cual se le van haciendo mejoras, hasta obtener un diseño final, su calidad se evalúa durante su elaboración, se deben minimizar los errores conceptuales. Un diseño con calidad produce un sistema robusto que tiene previsto como manejar los casos no exitosos.

Calidad en el Código para codificar con calidad es necesario apegarse a las *Reglas de Codificación* establecidas por la empresa u organización responsable del sistema. De esta manera se promueve que el código sea legible. La *Lectura de Código* entre colegas ayuda a detectar a tiempo muchos errores que disminuyen el tiempo que se ocupará más tarde en probar y corregir.

En el mapa conceptual de la Figura 3-13 se resumen los tipos de pruebas en función de su alcance.

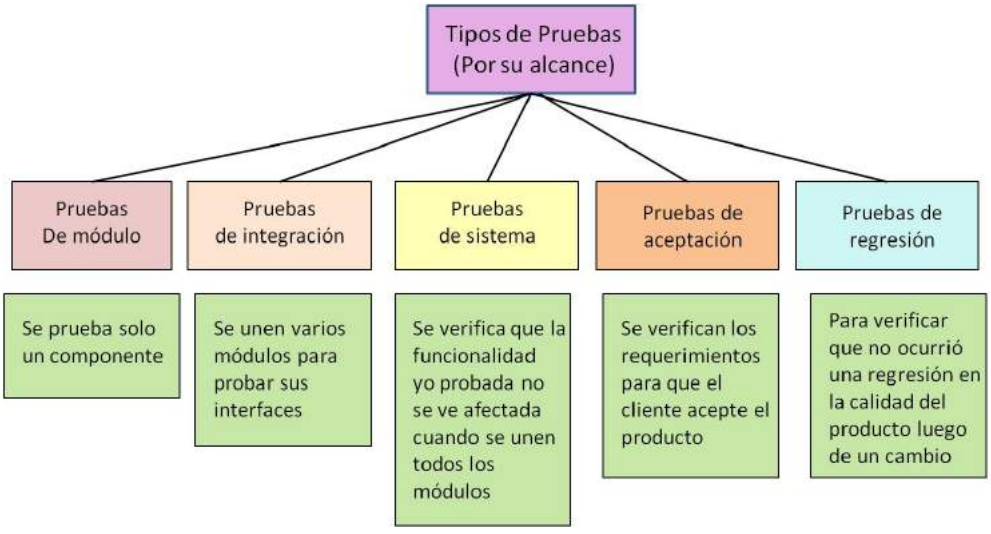


Figura 3-13: Tipos de pruebas por su alcance

Plan de Pruebas es el documento en el que se describe el alcance, la estrategia, los recursos necesarios y la agenda o calendarización de todas las pruebas que se llevarán a cabo en un proyecto.

*Diseño de Pruebas e*s un documento en el que se describe un grupo de casos de pruebas relacionadas entre sí. Cada documento de este tipo debe estar referenciado en el plan de pruebas.

*Caso de prueba e*s un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y pos-condiciones de ejecución, desarrollados con un objetivo particular o condición de prueba.

*Procedimiento de Prueba d*escribe los pasos a seguir para evaluar cierta funcionalidad de un sistema o subsistema. Contiene las instrucciones detalladas para la configuración, ejecución y evaluación de los resultados de uno o varios casos de prueba.

Bitácora de pruebas en este documento se registran brevemente todos los eventos en orden cronológico que suceden durante las pruebas.

Reporte de entrega de módulos: Identifica los módulos Software que han sido entregados al departamento de prueba.

Reporte de error de prueba (*T-record*) es un documento en el que se registra un evento ocurrido durante la ejecución de alguna prueba, y que requiera investigarse. Normalmente se trata de reportes de *fallas* en el resultado de una prueba lo que indica la posibilidad de que existan *defectos* en el SW.

Resumen de pruebas (*Test-Log*) resume los resultados de todas las pruebas para poder evaluarlos.

3.7 Cuestionario

1. ¿Cuáles son las actividades básicas de un proceso de desarrollo de software?
2. Describe brevemente en qué consisten estas actividades y cuáles son los principales productos de cada una de ellas.
3. ¿En qué consisten las pruebas del software?
4. ¿Cuáles son las actividades básicas del proceso de pruebas?
5. Describe brevemente en qué consiste cada una de las actividades del proceso de pruebas.
6. ¿Qué son las pruebas de aceptación?
7. ¿Qué son las solicitudes de cambio?
8. ¿Qué aspectos se toman en cuenta para darle calidad al diseño?
9. ¿Qué aspectos se toman en cuenta para darle calidad al código?
10. ¿En qué consisten las técnicas de prueba de caja negra y de caja blanca?
11. ¿Cómo se clasifican las pruebas en función de su alcance?
12. Explica brevemente en qué consiste cada uno de los tipos de prueba del punto anterior.
13. Explica en qué consiste cada uno de los siguientes documentos:

Plan de Pruebas
Diseño de Pruebas
Caso de prueba
Procedimiento de Prueba
Bitácora de pruebas
Reporte de entrega de módulos
Reporte de error de prueba (*T-record*) *Resumen de pruebas* (*Test-Log*)

4. Prácticas de Laboratorio

4.1 Objetivo general de las prácticas de laboratorio

- Aplicar los conocimientos adquiridos en teoría a un proyecto de software real.

4.2 Presentación de QualiTeam

QualiTeam es una aplicación web que ayuda al control de la documentación asociada a diferentes proyectos siguiendo los principios y procedimientos de la calidad de la Ingeniería del Software. Es una herramienta para que los alumnos que cursen la Unidad de Enseñanza Aprendizaje "Calidad y Pruebas" lleven a cabo sus prácticas. Además, es un sistema que facilita el desarrollo de software en equipo. Una vez que lo conozcan y sepan operarlo, profesores y alumnos podrán utilizarlo para coordinar los trabajos en equipo de proyectos terminales y de otros proyectos de software de la licenciatura de Ingeniería en Computación. QualiTeam ayuda a que los alumnos asimilen los conceptos básicos de la calidad y de la documentación de pruebas.

QualiTeam está disponible en la siguiente liga:
http://qualiteam.cua.uam.mx:8080/QualiTeam/

4-1: Página de acceso a ualiTeam

Con QualiTeam, el profesor puede ver el registro de las contribuciones individuales de cada participante, lo que le permite identificar desviaciones e intervenir de manera oportuna para orientar a los alumnos en sus esfuerzos, promoviendo así una participación más equitativa durante el proceso de enseñanza-aprendizaje. Esto también ayuda a que el profesor haga una evaluación más detallada y justa de los integrantes de un equipo.

4.3 Práctica 1.- Administración de un proyecto de Software con QualiTeam

4.3.1 Objetivos particulares de la práctica 1

- Comprender el funcionamiento general de QualiTeam.
- Ingresar al sistema la documentación de un proyecto de software real.

4.3.2 Ingreso al sistema

QualiTeam se encuentra en http://qualiteam.cua.uam.mx:8080/QualiTeam/, para ingresar al sistema por primera vez, es necesario registrarse seleccionando el botón "*Registrar*" de la página de ingreso. Después hay que proporcionar los datos solicitados en la interfaz de la Figura 4-2.

Figura 4-2: Interfaz de registro de usuario de QualiTeam

Cuando el usuario ya está registrado, sólo hay que proporcionar correctamente la cuenta y la contraseña para acceder a la página principal (Home) de QualiTeam, que se ilustra en Figura 4-3. En esta página se despliega la lista de todos los proyectos registrados en el sistema.

Figura 4-3: El "Home" de QualiTeam muestra la lista de proyectos registrados

El usuario puede seleccionar un proyecto de los existentes, o crear uno nuevo. Cuando crea un proyecto nuevo, adquiere el rol del "*líder*" de ese proyecto. Un líder puede editar su proyecto para modificar su nombre y la lista de los integrantes. Cualquier usuario puede ver los documentos del proyecto.

4.3.3 Creación y edición de un proyecto

El líder de cada equipo deberá registrar su proyecto en QualiTeam, el procedimiento es el siguiente.

- Seleccionar el botón "*Crear Nuevo*" en la interfaz de la Figura 4-3
- Se desplegará un renglón adicional para el nuevo proyecto creado y el usuario que lo creó es el líder del proyecto creado.
- Para poner el nombre y los integrantes del proyecto, el líder deberá seleccionar "Editar Proyecto" en la interfaz de la Figura 4-3.
- Cada vez que se requiera hacer cambios en el nombre y/o en los integrantes del proyecto, el líder podrá editar el proyecto por medio del botón "Editar Proyecto".

En la Figura 4-4 se muestra la interfaz de QualiTeam para editar un proyecto.

Figura 4-4: Edición de un proyecto. Nota: *Sólo el líder tiene acceso a la edición del proyecto.*

4.3.4 Consulta de los proyectos registrados

Todos los usuarios de QualiTeam pueden consultar los documentos de cualquiera de los proyectos. Sin embargo, sólo los integrantes de los proyectos podrán crear documentos y ser revisores de los documentos del proyecto. La interfaz de usuario de la Figura 4-5 se accede pulsando el botón "Documentos Proyecto" de la Figura 4-3, y muestra los documentos asociados a un proyecto.

Figura 4-5: Lista de documentos asociados a un proyecto (el usuario es integrante del proyecto)

En la Figura 4-5 se puede apreciar que el acceso a los subsistemas de Pruebas, Reportes de Error y Solicitudes de Cambio se encuentra en el extremo izquierdo. En la siguiente sección se especifica el manejo de los documentos asociados a un proyecto.

4.3.5 Documentos asociados a un proyecto

4.3.5.1 Los documentos de inicio y los demás documentos

Cuando se crea un proyecto, se generan automáticamente los *documentos de inicio,* que son: el Plan del Proyecto, la *especificación de requerimientos*, el documento de Diseño y el Plan de Pruebas. Es importante hacer notar que en QualiTeam, cuando se "genera" un documento, significa que se crea un nuevo registro del documento en la base de datos. Sin embargo, será necesario subir posteriormente el contenido del documento para que el documento en sí (en pdf) quede guardado en el servidor.

Con el botón "Crear Nuevo" de la interfaz en la Figura 4-5 se puede generar un nuevo documento dentro del proyecto seleccionado. Cada vez que se genera un documento nuevo, aparece un renglón adicional con la identidad y versión del documento y el estatus "en creación". Para agregar la información adicional del documento será necesario "*Editar el detalle*", como se explicará más adelante.

Cuando el usuario no es integrante de un proyecto, se presenta la interfaz de la Figura 4-6 en la que se puede apreciar que no aparece el botón "Crear Nuevo" documento, ya que sólo los integrantes del proyecto pueden crear documentos en éste.

Figura 4-6: Documentos asociados al proyecto (el usuario no es integrante)

4.3.5.2 Ver Detalle de un documento

Lider o autor: Con el botón "Ver Detalle" de un documento de la lista de documentos del proyecto seleccionado, se accede a la interfaz de la Figura 4-7, en la que se despliega la siguiente información: título del documento, autor, estatus (en creación, en revisión, revisado, aceptado, rechazado), esfuerzo y lista de revisores. Las funciones del botón "ver comentarios" se explicarán más adelante.

Figura 4-7: Interfaz que muestra los detalles de un documento (*lider y autor*)

4.3.5.3 La edición del detalle de un documento

Cuando el usuario es el autor del documento que está consultando, o es el líder del proyecto, aparece también el botón "Editar Propiedades", como en la interfaz de la Figura 4-7, el cual conduce a la interfaz de la Figura 4-8.

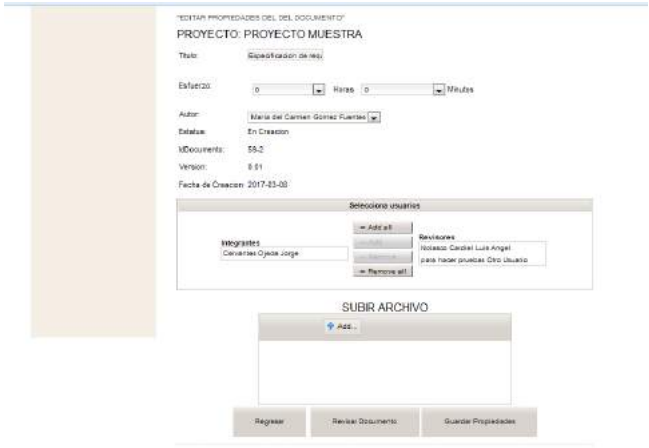


Figura 4-8: Interfaz para editar los detalles del documento

En esta interfaz el usuario puede modificar los campos. Para que las modificaciones queden registradas es necesario seleccionar el botón "Guardar propiedades". Cuando se guardan las propiedades se despliega la interfaz "Ver Detalles del Documento" con las modificaciones hechas.

Mediante el botón +Add… en "Subir Archivo" se puede subir al servidor una nueva versión del documento, la cual reemplazará a la que estaba almacenada previamente en el servidor. Al principio, cuando aún no se ha subido un documento, no aparece el botón de "Bajar Contenido" (como en la Figura 4-7). Pero una vez que ya hay documento en el servidor, entonces aparece el botón "Bajar Contenido" (como en la Figura 4-9).

Otros usuarios: A los usuarios que no son el autor del documento, ni el líder del proyecto, se les presenta la interfaz de la Figura 4-9 en que no está presente el botón "Editar Propiedades". Sin embargo, para ellos es posible bajar el contenido del documento, ver las propiedades y los comentarios del documento.

Figura 4-9 Interfaz que muestra los detalles de un documento (otros usuarios)

Con el botón "bajar contenido" se puede descargar el contenido del documento a la computadora del usuario.

4.3.6 Actividades de la práctica

a) Formar equipos. Cada equipo desarrollará un proyecto de software. Normalmente es el proyecto que están desarrollando en la UEA "Proyecto de Software II". Los miembros del equipo deberán registrarse en QualiTeam.
b) Designar un líder de proyecto por cada equipo, y un autor para cada documento del proyecto. Cada integrante deberá ser autor de al menos un documento.
c) El líder deberá dar de alta el proyecto del equipo en QualiTeam, y designar a los integrantes del equipo.
d) Una vez dados de alta en el proyecto, cada integrante deberá crear su documento asignado, si es uno de los documentos de inicio, el Líder debe asignarlo como autor para que pueda editarlo.

4.4 Práctica 2.- El Proceso de Revisión entre Colegas

4.4.1 Objetivos particulares de la práctica 2

- Aplicar el procedimiento de revisión entre colegas de productos de software.
- Fomentar la crítica respetuosa y constructiva entre los compañeros de equipo.

4.4.2 El ciclo de Revisión de un documento

Pondremos en práctica el ciclo de revisión entre colegas descrito en la sección 2.3.1. Cada alumno del equipo es responsable de uno de los documentos del proyecto que están desarrollando. Se nombra a un líder, quien será el responsable de poner los documentos en estatus de "Aceptado" o "Rechazado" cuando finaliza el ciclo.

4.4.2.1 La lista de revisores y el cambio de estatus

Cuando el autor termina de escribir un documento, debe editar la lista de revisores para indicar quienes serán los responsables de leer su documento y retroalimentar con comentarios. El autor y el líder pueden modificar la lista de revisores del documento en la interfaz de "Editar Propiedades" (Figura 4-8). Para enviarlo a revisión, el autor debe cambiar el estatus del documento a "En Revisión" por medio del botón "Revisar Documento" en la interfaz "Editar Propiedades". Una vez que el documento queda en estatus "En Revisión", aparecen nuevos botones en la interfaz de "Ver Detalle" para el autor y el líder, como se muestra en la Figura 4-10.

Figura 4-10: Las opciones para: crear nueva versión, aceptar o rechazar un documento

Cuando se crea un documento por primera vez, la versión es automáticamente la 0.01. La siguiente versión se incrementará en 0.01. Para crear una nueva versión siempre será necesario poner en estatus "En Revisión" el documento actual. Además, cuando se genera la nueva versión, el estatus del documento actual pasa automáticamente de "En Revisión" a "Revisado", como se ilustra en la Figura 4-11.

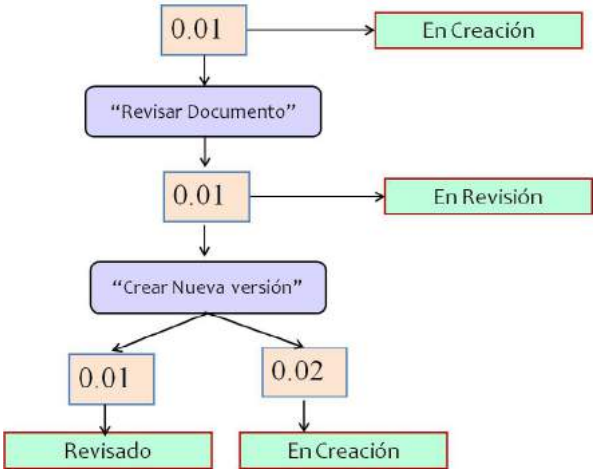


Figura 4-11: Creación de la nueva versión de un documento

Una vez que el autor pone su documento "En Revisión", los miembros del equipo deberán leer los documentos en los que participan como revisores. Los comentarios para mejorar el documento se introducen en el sistema como se explica en la sección titulada "Los comentarios de un documento".

El líder del proyecto pone el documento en estatus "Aceptado" o "Rechazado" cuando da por finalizado el ciclo de revisión entre colegas. No se puede editar un documento que está en estatus "Aceptado" o "Rechazado". La creación de nuevas versiones de documentos con estatus "Aceptado" se explica en la práctica "Reportes de Error".

4.4.2.2 Los comentarios de un documento

El sistema de comentarios de QualiTeam se accede mediante el botón "Comentarios" de la interfaz "Ver Detalle" (Figura 4-7). Cualquier usuario puede ver los comentarios de un documento al seleccionar el botón "Comentarios", sin embargo, sólo los miembros de la lista de revisores, el autor y el líder tienen permiso para agregar comentarios, por lo que sólo ellos tienen acceso a la interfaz de la Figura 4-12, que funciona de la siguiente forma.

Figura 4-12: Interfaz para ver e introducir comentarios

En la columna del lado izquierdo se despliega la lista de hilos de conversación, se debe agregar un nuevo hilo para cada tema, esto se hace con el botón "Nuevo Hilo". Al seleccionar uno de los hilos con el mouse, en la columna del lado derecho se despliega la lista de comentarios asociada al hilo de conversación seleccionado. Para agregar un comentario hay que escribirlo en el campo de texto de la parte inferior y posteriormente guardarlo con el botón "Agregar". Cuando se oprime este botón el comentario aparece en la lista de comentarios y desaparece del campo de texto.

Tanto el autor como el líder y los revisores, pueden cerrar un hilo de conversación con el botón "Cerrar Hilo". Nadie puede agregar más comentarios en un hilo que está cerrado. Si alguien desea hacer más comentarios deberá crear un nuevo hilo. Con el botón "Regresar" se despliega nuevamente la interfaz de "Ver Detalles del Documento".

4.4.2.3 Atención de los comentarios

Después de que el autor atendió a todos los comentarios, crea una nueva versión decimal del documento con el botón "Crear Siguiente Versión" que está en la interfaz "Ver Detalle" (Figura 4-10). Nótese que en esta interfaz sólo aparece al autor y al líder, los demás usuarios no pueden crear nuevas versiones del documento.

La nueva versión debe contener los comentarios atendidos. Cuando el autor considera que el documento está completo, comienza nuevamente al proceso de revisión, pero ahora de la nueva versión. El ciclo continúa hasta que ya no hay más comentarios o cuando el líder decide terminar el proceso y aceptar o rechazar el documento.

4.4.3 Actividades de la práctica

a) Cada autor deberá subir su documento en versión .pdf, asignar como revisores a los integrantes del equipo y someterlo a revisión.
b) Todos los integrantes del equipo revisarán los documentos de sus compañeros e ingresarán sus comentarios en QualiTeam.
c) Cada autor atenderá los comentarios hechos a su documento y creará una nueva versión que contenga los comentarios atendidos.
d) Se llevará a cabo el proceso de revisión cíclicamente hasta que el autor y el líder estén de acuerdo en finalizarlo. Entonces se pondrá el documento en "Aceptado".

4.5 Práctica 3.- El Control de Pruebas

4.5.1 Objetivo particular de la práctica 3

- Administrar la documentación de pruebas de software según el estándar IEEE 829.

4.5.2 La Documentación de pruebas

4.5.2.1 Plan de pruebas del proyecto

El plan de pruebas del proyecto es uno de los cuatro documentos que se crean automáticamente cuando se crea un proyecto, como se aprecia en la Figura 4-13. El plan de pruebas contiene la agenda de las actividades de prueba, objetivos y requerimientos de las pruebas, personal que va a hacer las pruebas, recursos SW y HW, etc. (ver 3.5.1). El líder debe subir el contenido del documento "Plan de Pruebas" al proyecto del equipo.

Figura 4-13: Entrada al subsistema de control de pruebas

Para trabajar con el resto de los documentos de pruebas, hay que seleccionar el subsistema de control de pruebas, como se indica en la Figura 4-13. El "home" del subsistema de pruebas se presenta en la Figura 4-14.

El "Home" de pruebas contiene los botones para acceder al control de: "Procedimientos de Prueba", "Test Logs" y "Diseños de Pruebas". Los Casos de Prueba se encuentran dentro de los Diseños de Pruebas.

Figura 4-14: Home del subsistema pruebas

Cada uno de los autores de los demás documentos de pruebas: test logs, procedimientos, diseños y casos de prueba deberá crear en QualiTeam el o los documentos que le correspondan de acuerdo a los procedimientos indicados en las siguientes sub-secciones.

4.5.2.2 Procedimiento de Pruebas

Un procedimiento de pruebas describe los pasos a seguir para evaluar cierta funcionalidad de un sistema o subsistema. Contiene las instrucciones para configurar, ejecutar y evaluar uno o varios casos de prueba (ver 3.5.4). En la Figura 4-14 se muestra la interfaz que enlista los procedimientos de prueba creados para el proyecto en el que se está trabajando. Al principio no existen procedimientos de prueba en el proyecto. Para generar un procedimiento de pruebas hay que seleccionar "Crear Nuevo", y después con "Ver Detalle"/"Editar Propiedades" se pueden editar las propiedades del documento (título, lista de distribución, subir el documento al servidor,…).

Figura 4-15: Procedimientos de Prueba del proyecto

El autor de un Procedimiento de Pruebas deberá dar de alta un nuevo documento. Después poner el título y demás propiedades. Cuando considere que el documento está listo para su revisión deberá subirlo al servidor en formato..pdf. Actualizar la lista de revisores y ponerlo "En Revisión". En la Figura 4-16 se muestra un ejemplo de dos procedimientos de prueba ya creados, el primero de ellos ya está "En Revisión".

Figura 4-16: Dos procedimientos de prueba en el proyecto

4.5.2.3 Test Logs

Los resultados de los casos de prueba se reportan en un Resumen de Pruebas (Test Data Summary, ver 3.5.9). En la Figura 4-18 se muestra la interfaz que enlista los Test Logs creados para el proyecto en el que se está trabajando. Al principio no existen Test Logs en el proyecto. Para generar un Test Log hay que seleccionar "Crear Nuevo", y después con "Ver Detalle"/"Editar Propiedades" se pueden editar las propiedades del documento (título, lista de distribución, subir el documento al servidor,…).

El autor de un Test Log deberá dar de alta un nuevo documento. Después poner el título y demás propiedades. Cuando considere que el documento está listo para su revisión deberá subirlo al servidor en formato .pdf. Actualizar la lista de revisores y ponerlo "En Revisión". La Figura 4-18 muestra un ejemplo de un Test Log con varias versiones y de otro Test Log que está "En Creación".

Figura 4-17: Un Test Log con varias versiones y otro "En Creación"

4.5.2.4 Diseños de Pruebas

El diseño de pruebas incluye la información referente al grupo de casos de prueba que éste abarca (ver 3.5.2). En la Figura 4-18 se muestra la interfaz que enlista los diseños de pruebas creados para el proyecto en el que se está trabajando. Al principio no existen diseños de pruebas en el proyecto. Para generar un diseño de pruebas hay que seleccionar "Crear Nuevo", y después con "Ver Detalle"/"Editar Propiedades" se pueden editar las propiedades del documento (título, lista de distribución, subir el documento al servidor,…).

El autor de un Diseño de Pruebas deberá dar de alta un nuevo documento. Después poner el título y demás propiedades. Cuando considere que el documento está listo para su revisión deberá subirlo al servidor en formato .pdf. Actualizar la lista de revisores y ponerlo "En Revisión". La Figura 4-18 muestra un ejemplo de diseños de pruebas ya creados.

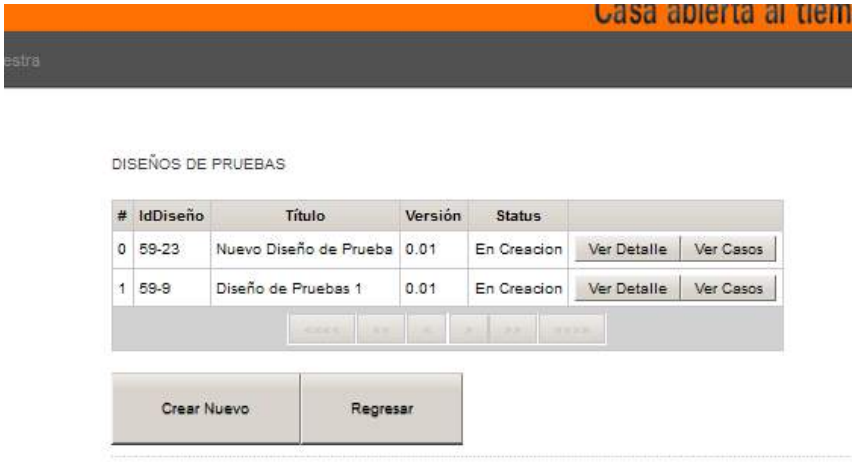


Figura 4-18: Diseños de Pruebas de un proyecto

La interfaz de Diseño de Pruebas contiene un botón adicional para cada documento, que es el de "Ver Casos". Esto se debe a que cada caso de prueba debe estar asociado a un diseño de pruebas.

4.5.2.5 Casos de Prueba

El caso de prueba contiene las condiciones, valores de entrada y resultados esperados con el objetivo de probar un caso particular (ver 3.5.3). Al seleccionar "Ver Casos" en un Diseño de Pruebas, se despliegan todos los casos de prueba asociados a ese diseño, como se puede ver en la Figura 4-19.

Figura 4-19: Casos de Prueba asociados a un Diseño de Pruebas

El autor de un Caso de Prueba debe seguir el mismo procedimiento descrito para los otros documentos de pruebas (Procedimientos de Prueba, Test Logs, Diseños de Pruebas).
Pero, además, tiene la opción de asociar un Procedimiento de Prueba a un Caso de Prueba. En la Figura 4-19 puede observarse que el "Caso de Prueba 1" ya tiene un Procedimiento de Prueba asignado, mientras que los otros dos casos no. Para asignar un procedimiento a un caso de prueba se usa el botón "Seleccionar Procedimiento", el cual conduce a la interfaz que contiene la lista de Procedimientos de Prueba del proyecto (Figura 4-20).

Figura 4-20: Procedimientos de Prueba de un proyecto

Con el botón "Asignar Proc." se asigna el Procedimiento de Prueba seleccionado. En este ejemplo se eligió el Procedimiento de Prueba 1 (Id: 59-7). Al al elegir "Asignar Proc" se despliega nuevamente la lista de casos de prueba y, como se aprecia en la (Figura 4-21), en la columna "IdProcedimiento" aparece la identidad del Procedimiento de Prueba que se asignó al Caso de Prueba.

Figura 4-21: Caso de Prueba con un Procedimiento de Prueba asignado

4.5.3 Actividades de la práctica

a) Crear uno o dos *Procedimientos de Prueba* para el proyecto del equipo y someterlos a un proceso de revisión hasta que estén "Aceptados".
b) Crear uno o dos *Diseños de Pruebas* para el proyecto del equipo y someterlos a un proceso de revisión hasta que estén "Aceptados".
c) Crear varios *Casos de Prueba* para un *Diseño de Pruebas*, asignarles un *Procedimiento de Prueba* y someterlos a un proceso de revisión hasta que estén *Aceptados*.

> Nota: A estas alturas del curso, lo más probable es que los equipos aún no tengan la implementación de su proyecto terminada. Como un Test Log es el resultado de las pruebas que se le hacen a un software que ya está funcionando, no se crearán Test Log en esta práctica.

4.6 Práctica 4.- Los Reportes de Error

4.6.1 Objetivo particular de la práctica 4

- Aplicar el procedimiento de reportes de error a un proyecto real.

4.6.2 El subsistema de Reportes de Error de QualiTeam

Cuando un documento se encuentra en el estatus de *Aceptado*, ya no se le deben hacer cambios. Entonces, en lugar de modificarlo, el mecanismo que se usa para corregirlo es generar *Reportes de Error*. Se genera un reporte de error para cada *defecto* encontrado, y cuando se considere apropiado, se crea una nueva versión del documento, la cual atiende a todos los *Reportes de Error* generados para éste.

En la Figura 4-22 se ilustra como acceder al subsistema de "Reportes de Error" de los documentos del proyecto seleccionado en QualiTeam.

Figura 4-22: Acceso a Reportes de Error

4.6.3 Creación de Reportes de Error y su proceso de revisión

El subsistema de Reportes de Error muestra la lista de Reportes generados para los documentos aceptados de un proyecto en particular. En el ejemplo de la Figura 4-23 se muestran dos Reportes de Error para el documento de Diseño, y uno para el *Documento de Requerimientos*. El proceso de revisión de los Reportes de Error es el mismo que para cualquier documento, es decir, primero tiene estatus "En Creación", y después se pone "En Revisión". Al crear la siguiente versión, la anterior queda en "Revisado". También existe la opción de poner un Reporte de Error en "Aceptado" o "Rechazado".

Figura 4-23: Lista de Reportes de Error de un proyecto

Cada vez que se selecciona "Crear Reporte de Error" se despliega la interfaz de la Figura 4-24, la cual muestra únicamente los documentos del proyecto que están aceptados, ya que no tiene sentido hacer un Reporte de Error sobre un documento que aún no ha sido aceptado. Con el botón "Seleccionar Documento", se crea el nuevo reporte de error asociado al documento base que se seleccionó.

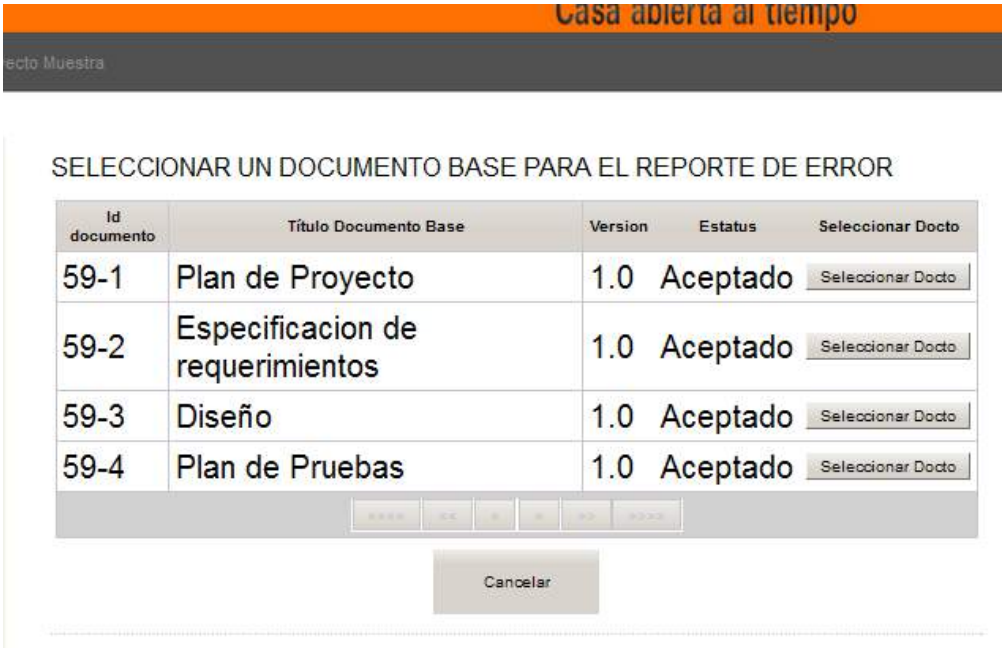


Figura 4-24: Selección de un documento base para el Reporte de Error en creación

4.6.4 Atención a los reportes de error: creación de una nueva versión del producto

Hasta ahora hemos trabajado con la versión decimal de un documento: 0.01, 0.02, y subsecuentes hemos visto que cuando aceptamos un documento éste pasa a una versión entera (1.0). Cuando se incorporan los reportes de error a una nueva versión, primero se genera su siguiente versión decimal, por ejemplo 1.01, y se somete al proceso de revisión, con lo cual se generan las siguientes versiones decimales: 1.02, 1.03, y subsecuentes. Cuando finalmente se aprueba la siguiente versión, QualiTeam asigna automáticamente la siguiente versión entera, que en este ejemplo sería la 2.0.

La Figura 4-25 ilustra la manera en la que se generan las nuevas versiones, decimales y enteras de un documento.

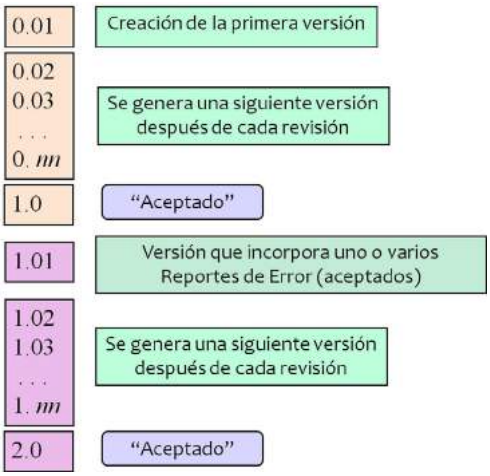


Figura 4-25: Secuencia de versiones decimales y enteras de un documento

4.6.5 Actividades de la práctica

a) Generar Reportes de Error sobre versiones aceptadas de los documentos del proyecto. El Reporte de Error debe señalar algún *defecto* encontrado en el documento base.
b) Crear la nueva versión del documento base que incorpore los Reportes de Error generados.
c) Iniciar el proceso de revisión de la nueva versión hasta generar la nueva versión aceptada.

4.7 Práctica 5.- Las Solicitudes de Cambio

4.7.1 Objetivo particular de la práctica 5

- Aplicar el procedimiento de solicitudes de cambio a un proyecto real.

4.7.2 El subsistema de Solicitudes de Cambio de QualiTeam

Las Solicitudes de Cambio son documentos que describen una modificación que debe hacerse sobre un documento ya aceptado, al cual se le llama *documento base*. Esta modificación no tiene como origen un defecto en el documento base, sino que surge de la necesidad de hacer

mejoras para una nueva versión del proyecto base o adaptaciones para un nuevo proyecto. Y son precisamente estas mejoras o adaptaciones las que se incluyen en las Solicitudes de Cambio. En la Figura 4-26 se muestra el acceso a las Solicitudes de Cambio de QualiTeam.

Figura 4-26 Acceso a las Solicitudes de Cambio

4.7.3 Creación y edición de Solicitudes de Cambio

El subsistema de Solicitudes de Cambio muestra la lista de Solicitudes de Cambio generadas para el documento de un proyecto base seleccionado. En el ejemplo de la Figura 4-27, el proyecto actual es QualiTeam 2.0, el subsistema de solicitudes de cambio contiene dos Solicitudes de Cambio, la 5-5 está hecha sobre el documento de Especificación del Proyecto QualiTeam 1.0 y la 5-6 sobre el de Diseño de QualiTeam 1.0.

El proceso de revisión de las Solicitudes de Cambio es el mismo que para cualquier documento, es decir, primero tiene estatus "En Creación", y después se pone "En Revisión". Al crear la siguiente versión, la anterior queda en "Revisado". También existe la opción de poner una Solicitud de Cambio en "Aceptado" o "Rechazado".

Figura 4-27: Lista de las Solicitudes de Cambio en un proyecto

Cada vez que se selecciona “Crear Solicitud de Cambio” se despliega la interfaz de la Figura 4-28, la cual muestra la lista de todos los proyectos, excepto el proyecto actual (en nuestro ejemplo QualiTeam 2.0) . Cuando se selecciona un proyecto base, a la derecha se despliegan sus documentos (únicamente los aceptados). Con el botón “Seleccionar Docto”, se crea la nueva Solicitud de Cambio asociada al documento del proyecto base que se seleccionó.

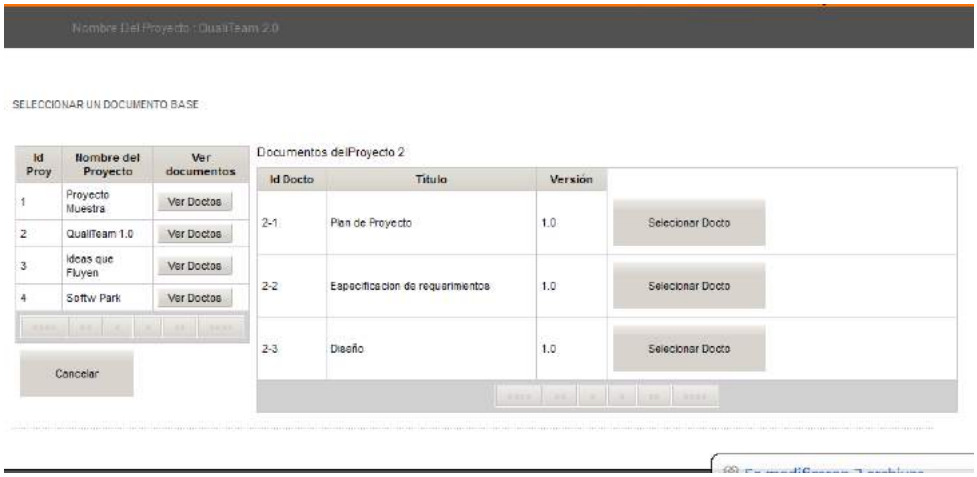


Figura 4-28: Proyectos Base con sus Documentos (aceptados)

4.7.4 Ejemplo de Solicitud de Cambio

Una buena forma de comprender lo que es una Solicitud de Cambio es con un ejemplo ilustrativo. En esta sección incluimos una Solicitud de Cambio para el proyecto QualiTeam, el cuál será incorporado en la versión 2.

Es importante destacar que una Solicitud de Cambio no debe hacerse sobre documentos del proyecto en el que se está trabajando. Por eso creamos un nuevo proyecto llamado QualiTeam 2.0, y en este nuevo proyecto se incorporarán varias mejoras al proyecto QualiTeam 1.0. Una de las mejoras es la que se describe en la siguiente Solicitud de Cambio, a la que hemos puesto el identificador 5-6.

Solicitud de Cambio “5-6”

Autor: Xyz

Proyecto Actual: QualiTeam 2.0

Proyecto Base: QualiTeam 1.0

Documento Base: Diseño

TITULO: Agregar la opción de consultar las propiedades de un proyecto

Descripcion de la mejora:

En la página principal de QualiTeam 1.0: *AdministradorProyectos.xhtml*, se presenta el botón “Editar Proyecto”, el cual es accesible únicamente para el líder del proyecto. Así, los demás usuarios no pueden consultar quienes son los integrantes de cada proyecto. Con QualiTeam 2.0 será posible que cualquier usuario del sistema tenga acceso a la información de un proyecto.

Descripción del diseño que cumple con el nuevo requerimiento
Diseño de alto nivel
El Diagrama de Transición entre Interfaces de Usuario (DTIU) que cumple con el nuevo requerimiento se ilustra en la Figura 1.

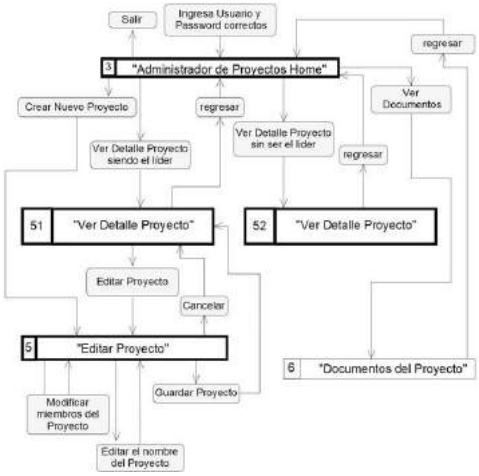


Figura 1: *DTIU para la consulta y edición de las propiedades de un proyecto*

AdministradorProyectos.xhtml tiene ahora el botón "Ver Detalle Proyecto" que redirecciona a la página *DetalleProyecto.xhtml*, con su correspondiente *managed bean* asociado que se llamará *DetalleProyecto.java*.

DetalleProyecto.xhtml contendrá exactamente los mismos campos que *EdicionDelProyecto.xhtml* pero éstos no serán editables. Para que se dibuje el botón de "Editar Proyecto" en la interfaz el usuario deberá ser el líder del proyecto seleccionado.

Diseño detallado

El diagrama de secuencia correspondiente al caso de uso "Ver Detalle del Proyecto se muestra en la Figura 2.

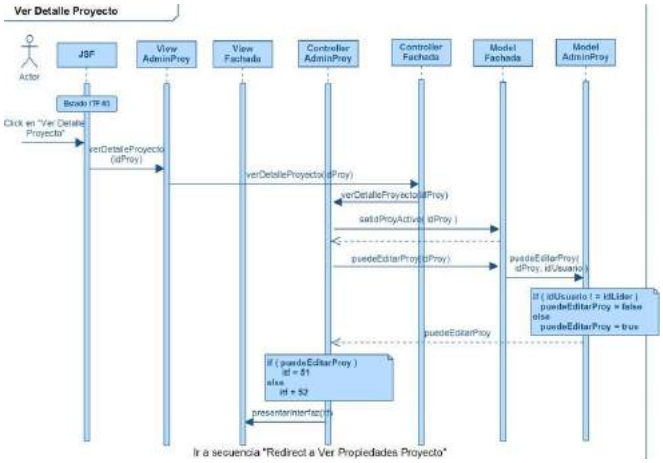


Figura 2: Diagrama de secuencia de "Ver detalle de un proyecto"

Y el diagrama de secuencia correspondiente al caso de uso “Redirect a ver propiedades de un proyecto” se muestra en la Figura 3.

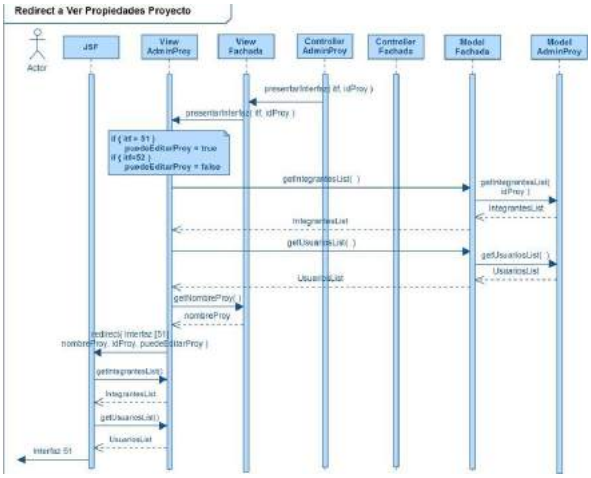


Figura 3: Diagrama de secuencia de “Redirect a ver propiedades de un Proyecto”

Módulos impactados:

- - ***DetalleProyecto.xhtml***: página nueva con la misma información que *EdicionDelProyecto.xhtml* pero no es editable. El botón para editar el proyecto se dibuja si la bandera `puedeEditarProy` está en “true”.
  <h:commandButton id=”Editar Proyecto” rendered=”#{DetalleProyecto.puedeEditarProy}” class=”button2” value=”Editar Proyecto” actionListener=”#{DetalleProyecto.editarProyecto()}”>
  </h:commandButton>
- ***DetalleProyecto.java***: managed bean nuevo asociado a la página xhtml. Contiene el método para atender el botón “Editar Proyecto” y otro para atender el botón “Regresar”. Pone la bandera `puedeEditarProy` en el estado que indica el view.
- ***AdministradorProyectos.xhtml***: cambiar el botón “Editar Proyecto” por el de “Ver Detalle Proyecto”, el cual redireccionará al método `verDetalleProy( )` de su managed bean asociado, y le envía como parámetro la identidad del proyecto seleccionado.
- ***AdministradorProyectos.java***: mover el método
`editarProyecto( String idProy )`
al managed bean *DetalleProyecto.java*, y sustituirlo por
`verDetalleProyecto( String idProy )`.
- ***EdicionDelProyecto.xhtml***: el botón “Regresar” dirá “Cancelar” y redirecciona a *DetalleProyecto.xhtml*. Eliminar el botón de “Salir”.

- ***ControllerAdminProy.java***: Agregar el método:

```
public void verDetalleProyecto( String
     idProy ) { elModel.setIdProyActivo(
     idProy );
     if( elModel.puedeEditarProy(
          idProy )){ elView.
          presentarInterfaz( 51 );
     }
     else
          elView.presentarInterfaz( 52 );
 }
```

- ***ViewFachada.java***: Agregar los casos para regresar de la interfaz 51, 52 en el método `presentarInterfaz()`.
- ***ViewAdminProy.java***: Agregar los casos para regresar de la interfaz 51, 52 en el método `presentarInterfaz()`. Para la interfaz 51 poner la bandera `puedeEditarProy` en true y para la 52 ponerla en false. Agregar getter para esta bandera.

```
case 51:

          puedeEditarProy = true;
          projectInfo = elModel.getInfoProyectoActivo();
          break;

case 52:

          puedeEditarProy = false;
          projectInfo = elModel.getInfoProyectoActivo();
          numInterfaz = 51;
          break;
```

- ***Interfaces.java***: (En el paquete ClasesQualiTeam), agregar la interfaz número 51 *"DetalleProyecto.xhtml"*.

Como puede apreciarse en el ejemplo que hemos presentado en esta sección, las solicitudes de cambio deben ser lo más específicas posibles para reducir el tiempo y los errores durante la etapa de codificación.

4.7.5 Actividades de la práctica

- Crear la siguiente versión del proyecto con el que se ha trabajado durante el curso. Por ejemplo, si se tiene ProyectoDelCurso 1.0 (proyecto anterior) crear ProyectoDelCurso 2.0 (proyecto nuevo). Los documentos principales de ProyectoDelCurso 1.0: Plan

del Proyecto, *especificación de requerimientos*, Diseño y Plan de Pruebas, deben estar en "Aceptado".

- Proponer una mejora al proyecto anterior (ProyectoDelCurso 1.0 en nuestro ejemplo). Y redactarla a manera de *Solicitud de Cambio*. Basarse en el ejemplo ilustrativo de la sección anterior. Determinar los documentos que se verán afectados por la mejora y escribir una solicitud de cambio por cada documento afectado. Por ejemplo, una para el de *especificación de requerimientos*, y otra para el de Diseño (si es que hay impacto en los dos documentos).
- Seleccionar el proyecto nuevo (ProyectoDelCurso 2.0 en nuestro ejemplo) y generar una Solicitud de Cambio para cada documento afectado, del proyecto anterior (ProyectoDelCurso 1.0 en nuestro ejemplo).
- Subir a QualiTeam la o las solicitudes de cambio correspondientes en su versión .pdf.

-

5. Glosario

A

Aseguramiento de la calidad: Prevención de la introducción de *defectos* en el Software.

Actividades de mantenimiento de software: Son las que se llevan a cabo una vez que ya se entregó el producto y está operación.

C

Implementación de software: Consiste en codificar lo que está indicado en el diseño. Durante la implementación se construyen la base de datos, las interfaces de usuario y todo el código que contiene la lógica del sistema.

Control de calidad del software: El control de calidad del software es el conjunto de actividades destinadas a la detección y corrección temprana de *defectos* en el sistema de software que está en producción.

D

Defecto: Es una inconsistencia del producto con sus *requerimientos*, y debe eliminarse. Un *defecto* produce una *falla*.

Diseño de software: Define los componentes, las interfaces entre los componentes, las interfaces de usuario, la gestión de las tareas y la gestión de los datos del sistema que se va a implantar. El diseño debe satisfacer los *requerimientos* especificados.

Documentación de pruebas: Son los documentos de un proceso de pruebas, los cuales se asocian de la siguiente manera: los Diseños de Prueba, los Procedimientos de Prueba y los Test Log están asociados a un Plan de Pruebas. Cada Diseño de Prueba contiene un grupo de Casos de Prueba. Cada Caso de Prueba se asocia a un Diseño de Prueba, pero también puede asociarse a uno o a varios Procedimientos de Prueba.

E

Especificación de requerimientos: Es un documento que define, de forma completa, precisa y verificable, los requisitos y el comportamiento u otras características, de un sistema o componente de un sistema.

Estándares de Calidad de la IEEE: Son los documentos que se deben seguir cuando se pretende cumplir con las normas de calidad, durante el desarrollo de proyectos de software.

F

Falla: Una *falla* de Software es un comportamiento inadecuado durante la ejecución, que provoca resultados erróneos.

I

Inspecciones: Es una técnica avanzada de control de calidad en la que se revisa la consistencia entre dos documentos y se coteja que se cumplan las *buenas prácticas*. Estas buenas prácticas deben estar bien definidas y documentadas.

P

Proceso de desarrollo de software: Es el conjunto estructurado de las actividades requeridas para realizar un sistema de software. Estas actividades son: especificación (análisis y definición de *requerimientos*), diseño, implementación (codificación), validación (pruebas) y mantenimiento.

Productos de un proyecto de software: Son documentos que impactan en el producto final, tales como la especificación de *requerimientos*, el diseño, los módulos de código y también los documentos que indirectamente afectan a éstos como reglas de documentación, reglas de codificación, descripciones de procedimientos, listas de pruebas, etc.

Proyecto de software: Es un esfuerzo temporal para crear un producto, servicio o resultado único. La naturaleza temporal de los proyectos indica un principio y un final definidos. El final se alcanza cuando se logran los objetivos del proyecto o cuando se termina el proyecto porque sus objetivos no se cumplirán o no pueden ser cumplidos, o cuando ya no existe la necesidad que dio origen al proyecto.

Pruebas del software: Consisten en verificar que un producto cumple con los requisitos para los que fue creado.

R

Reporte de error: Es un documento en el que se registra un evento ocurrido durante la ejecución de alguna prueba que requiera investigarse. Normalmente se trata de reportes de *fallas* en el resultado de una prueba lo que indica la posibilidad de que existan defectos en el SW. Un reporte de error debe contener: información de control, resumen, descripción e impacto.

Revisión entre colegas: Es una técnica para revisar un documento o código durante las fases tempranas de desarrollo. Tiene como objetivo la disminución del *costo* de corrección de *defectos* encontrados en la fase de pruebas.

S

Sistema de software: Es el conjunto de módulos software que se coordinan entre sí para brindar al usuario los servicios requeridos.

Solicitud de cambio: Es un documento en el cual se indican los cambios que se deben hacer a un producto base, ya sea un documento o un módulo software.

V

Versión nueva de un producto de software: Es un documento, o un módulo SW que se genera a partir de una versión anterior. En la nueva versión quedan atendidos comentarios (sólo para documentos), reportes de error, o una solicitud de cambio escritos para la versión anterior.

6. Bibliografía

Black, R., Managing the testing process. John Wiley & Sons, 2002.

Burnstein, I., *Practical software testing*: *a process-oriented approach*. Springer Science & Business Media, 2006.

Douglas, N Arnold, 2000, *The Explosion of the Ariane 5*
http://www.ima.umn.edu/~arnold/disasters/ariane.html

Glass, R.L. *Building quality software*. Prentice Hall PTR. USA, 1997.

Gilb T., Graham D., *Software Inspection*, Addison Wesley, 1993.

Godbole Nina S, *Software Quality Assurance*, *Principles and Practice,* Alpha Science International Ltd., Oxford U.K., 2005.

Horch, J. W. *Practical guide to software quality management*. Artech House Publishers. Norwood, USA, 2003.

Humphrey, W. S., Managing the software process. Addison-Wesley. 1989.

IEEE-STD-830-1998 *Recommended Practice for Software Requirements Specifications,* IEEE Computer Society.

IEEE-STD-730-2002 *Standard for Software Quality Assurance Plans*, IEEE Computer Society.

IEEE STD-829 *Standard for Software Test Documentation*, IEEE Computer Society, 16 December 1998.

Kaner, C., Falk, J., Nguyen, H. Q. *Testing computer software*, *2nd ed*. Wiley.USA, 1999.

Khan RA, Mustafa K, Ahson S I, *Software Quality*, *Concepts and Practices*, Alpha Science International Ltd., Oxford U.K., 2006.

Lamancha, B. P., *Proceso de Testing Funcional Independiente*. Tesis de Maestría, Universidad de la República, Montevideo, Uruguay, 2006.

O'Regan G., *A Practical Approach to Software Quality*, Springer-Verlag, Berlin, 2002.

Oskarsson, O., Glass, R.L. *An ISO 9000 approach to building quality software*- Prentice Hall. USA, 1995.

Patrick D. T. O'Connor. *Test engineering: a concise guide to cost-effective design, development and manufacture*. Wiley. England, 2001.

Pfleeger, Shari L. *Ingeniería de software. Teoría y práctic*a,1ª edición. Editorial Pearson Education, Buenos Aires, 2002.

PMI Guía de los fundamentos para la Dirección de Proyectos (Guía del PMBOK) Project Management Institute, 2008.

Pressman, Roger S. *Ingeniería del Software: Un enfoque práctico*, 7ª edición. Editorial McGraw Hill, 2010.

Schulmeyer, G., *Handbook of software quality assurance*, 4th Ed, Artech House, London, 2008.

Sommerville, Ian. *Ingeniería del Software*, Editorial Pearson, México, 2011.

Weitzenfeld, A. *Ingeniería de Software Orientada a Objetos con UML*, *Java e Internet*. Editorial Thomson, 2004.

Whittaker, J. A. *How to break software: A practical guide to testing.* Addison-Wesley. England, 2002.

7. Apéndice A

Ejemplo de Reglas de codificación

1.Declaración de variables alineadas. Ejemplo:

```
private final    ModelFachadaOp       elModel;
private final    ControllerLogin      elControllerLogin;
private fina     ControllerDoctos     elControllerDoctos;
private final    ControllerPruebas    elControllerPruebas;
```

2.Constantes con mayúsculas y alineadas. Ejemplo:

```
public          static        final   int    PRUEBAS          = 1;
public          static final  int     int    CHANGE _ REQUEST = 2;
public          static final  int     int    REPORTES _ ERROR = 3;
```

3.Parámetros dentro de paréntesis con un espacio después abrir paréntesis y antes de cerrarlos. Ejemplo:

```
public void verDoctosProy( String idProy ){
        elController.verDoctosProy( idProy );
}
```

4.El código no debe pasar de 80 columnas.

a.Cuando los parámetros de un método no caben en 80 columnas, usar la notación alternativa con un parámetro en cada renglón. Ejemplo:

```
public void guardarProyecto( ArrayList<Integrante> integrantesList,
                    String nombre ) {
          ...
}
```

b.Cuando el acceso a datos o métodos de un objeto ocupe más de 80 columnas, usar la notación alternativa con cambio de línea en el punto el cual va alineado con el primer acceso en la línea anterior. Ejemplo:

```
elController
 .guardarProyecto( integrantesList, nombre );
```

5.Alinear los signos de "=" cuando hay un grupo de asignaciones. Ejemplo:

```
elControllerLogin       = new ControllerLogin( this, elView, elModel );
elControllerDoctos      = new ControllerDoctos( this, elView, elModel );
elControllerPruebas     = new ControllerPruebas( this, elView, elModel );
```

6.Alineación en el switch:

a. cada `case` va en un renglón
b. el `break` va en un renglón aparte.
c. las instrucciones van todas alineadas.

Ejemplo:

```
switch( numInterfaz ){
  case 1:
  case 2:
  case 4: elViewLogin.presentarInterfaz( numInterfaz );
          break;
  case 3:
  case 5:
  case 6:
  case 7: elViewAdminProy.presentarInterfaz( numInterfaz,
                                                nombreProyActivo );
          break;

}
```

7.En expresiones matemáticas y asignaciones, dejar un espacio antes y después del signo igual y de los operadores. Por ejemplo:

```
a =       ( b + c - d ) / 2;     // Bien!
a=        (b+c-d)/2;             // Mal!
```

8. Apéndice B

Ejemplos de procedimientos de prueba

Ejemplo 1
Título: **Modificación de las propiedades de un documento**
Autor: María del Carmen Gómez Fuentes

Procedimiento de Prueba para modificar las propiedades de un documento

Descripción

- El usuario entra con el rol de líder del proyecto o es el autor del documento que selecciona.
- En una interfaz que contiene una lista de documentos, el usuario selecciona un documento y la opción "*Ver Detalle*".
- El estatus del documento debe ser "en creación" o "en revisión".
- El sistema deberá desplegar la interfaz #8 "Ver Detalle del Documento" con la opción "*Editar*".
- Al seleccionar el botón "*Editar*" de la interfaz #8 se deberá desplegar la interfaz #12 "Editar Propiedades" en la que se pueden modificar todos los datos excepto la identidad del documento y el autor.
- Se hacen modificaciones en algunas de las propiedades y al oprimir el botón "*Guardar*" se despliega la interfaz #8 "Ver Detalle del Documento" con las propiedades modificadas.

Ejemplo 2
Título: **Ver comentarios siendo autor, líder, o miembro de la lista de distribución**
Autor: María del Carmen Gómez Fuentes

Procedimiento de prueba para ver los comentarios de un documento siendo autor, líder, o miembro de la lista de distribución

Descripción

- El usuario es líder, autor o miembro de la lista de distribución del documento que selecciona.
- En una interfaz que contiene una lista de documentos, el usuario selecciona un documento y la opción "*Ver Detalle*".
- El estatus del documento debe ser "en revisión".

- Al seleccionar el botón "*Ver Comentarios*" de la interfaz #8 se deberá desplegar la interfaz #10 "Ver Comentarios" en la que se pueden ver los comentarios del documento. Aparecen los comentarios correspondientes al primer hilo. En caso de que no haya comentarios, todo está vacío.
- Aparece una opción para agregar un nuevo hilo.
- También aparece un campo de texto para capturar un nuevo comentario y agregarlo al hilo correspondiente.
- Además, se despliega un botón para la opción de "*Cerrar hilo*".
- Al seleccionar un hilo diferente del seleccionado se despliegan los comentarios del nuevo hilo seleccionado y su campo de texto para capturar uno nuevo.
- Al dar click en "*Regresar*" se despliega nuevamente la interfaz #8.

9. Apéndice C

Ejemplo de Diseño de Pruebas

Título: **Diseño de Pruebas para el subsistema**: ***Administrador de Proyectos***
Autor: María del Carmen Gómez Fuentes

Los casos para este diseño de pruebas son:

1.- Crear un nuevo proyecto
2.- Edición de un proyecto
    2.1.- Edición de los datos del Proyecto cuando el usuario es el líder
        2.1.1.- Modificar nombre
        2.1.2.- Modificar lista de integrantes
    2.2.- Edición de los datos del Proyecto cuando el usuario no es el líder

3.- Entrar a "Documentos de un Proyecto"

4.- Crear un documento nuevo en un proyecto
    4.1.- Crear documento siendo integrante del proyecto seleccionado
    4.2.- Crear documento sin ser integrante del proyecto seleccionado

5.- Generar Documentos de Inicio
    5.1.- Generar documentos de inicio siendo integrante del proyecto seleccionado
    5.2.- Generar documentos de inicio sin ser integrante del proyecto seleccionado

6.- Ir a un subsistema: pruebas, requerimientos de cambio, reportes de error
    6.1.- Ir y regresar al subsistema de Pruebas
    6.2.- Ir y regresar al subsistema de Requerimientos de Cambio
    6.3.- Ir y regresar al subsistema de Reportes de Error

7.- Salir del sistema

10. Apéndice D

Ejemplo de Casos de Pruebas

Título: **Casos de Prueba para el subsistema:** ***Administrador de Proyectos***
Autor: María del Carmen Gómez Fuentes

A continuación se describen los casos para el diseño de pruebas para el subsistema: *Administrador de Proyectos*.

1.- Crear un nuevo proyecto

Descripción.- En la interfaz #3 "Administrador de Proyectos Home", el usuario selecciona "*Crear Nuevo*" proyecto. Debe aparecer un renglón adicional en la lista de proyectos con el nombre del proyecto vacío.

Posteriormente el usuario selecciona "*Editar*" el Proyecto y se debe desplegar la interfaz #5 "Editar Proyecto". Después de introducir los datos del proyecto (parcial o totalmente), al seleccionar "*Guardar Proyecto*" en la interfaz #5 se despliegan los datos del proyecto, actualizados. Con click en el botón "*Regresar*" se debe desplegar la interfaz #3 "Administrador de Proyectos Home" y si el usuario añadió el nombre del proyecto, éste debe aparecer en el renglón correspondiente.

El sistema deberá generar y desplegar los cuatro documentos del inicio del proyecto, que son:

1.- Plan del proyecto
2.- Especificación de requerimientos
3.- Diseño
4.- Plan de pruebas

En la parte superior debe aparecer el nombre del proyecto actual.

2.- Edición de un proyecto

2.1.- Edición de los datos del Proyecto cuando el usuario es el líder

Descripción.- En la interfaz #3 "Administrador de Proyectos Home", el usuario selecciona "*Editar*" el Proyecto y el sistema pasa la interfaz #5 "Editar Proyecto". Después de introducir los datos del proyecto (parcial o totalmente), selecciona "*Guardar Proyecto*" en la interfaz #5. A continuación el sistema debe presentar nuevamente la interfaz #5 pero con los datos actualizados. Se deberán ejecutar dos pruebas para este caso: 2.1.1 y 2.1.2.

2.1.1.- Modificar nombre

Descripción.- En la interfaz #3 "Administrador de Proyectos Home", el usuario selecciona "*Editar*" un proyecto, debe pasar a la interfaz #5"Editar Proyecto". Cuando el usuario modifica el nombre del proyecto y da click en "*Guardar*", se despliega el nuevo nombre. Posteriormente al click en "*Regresar*", el sistema debe presentar nuevamente la interfaz #3"Administrador de Proyectos Home" con el nombre del proyecto modificado.

2.1.2.- Modificar lista de integrantes

Descripción.- En la interfaz #3 "Administrador de Proyectos Home", el usuario selecciona "*Editar*" un proyecto, debe pasar a la interfaz #5"Editar Proyecto". El usuario modifica la lista de integrantes del proyecto y da click en "*Guardar*", el sistema debe presentar nuevamente la interfaz #5 con la lista de integrantes modificada.

2.2.- Edición de los datos del Proyecto cuando el usuario no es el líder

Descripción.- En la interfaz #3 "Administrador de Proyectos Home", el usuario selecciona "*Editar*" el Proyecto y el sistema no debe permitir la edición del proyecto, indicando al usuario que se requiere ser el líder para poder editarlo.

3.- Entrar a "Documentos de un Proyecto"

Descripción.- En la interfaz #3 "Administrador de Proyectos Home", el usuario selecciona "*Documentos Proyecto*" y el sistema pasa la interfaz #6 "Documentos del Proyecto", en la que se deberá desplegar una lista con todos los documentos del proyecto seleccionado. En la parte superior debe aparecer el nombre del proyecto actual.

4.- Crear un documento nuevo en un proyecto

4.1.- Crear documento siendo integrante del proyecto seleccionado

Descripción.- En la interfaz #6 "Documentos del Proyecto", el usuario selecciona "*Crear Documento*", el sistema deberá desplegar un renglón adicional en la lista de los documentos del proyecto. En la parte superior debe aparecer el nombre del proyecto actual.

4.2.- Crear documento sin ser integrante del proyecto seleccionado

Descripción.- En la interfaz #6 "Documentos del Proyecto", el usuario selecciona "*Crear Documento*", el sistema deberá indicar que solo los integrantes del proyecto pueden crear documentos. No se modifica la lista de documentos del proyecto.

6.- Ir a un subsistema: pruebas, requerimientos de cambio, reportes de error

6.1.- Ir y regresar al subsistema de Pruebas

Descripción.- Aplicar el procedimiento de prueba "Acceso a los subsistemas", con el caso "Pruebas".

6.2.- Ir y regresar al subsistema de Requerimientos de Cambio

Descripción.- Aplicar el procedimiento de prueba "Acceso a los subsistemas", con el caso "Requerimientos de Cambio".

6.3.- Ir y regresar al subsistema de Reportes de Error

Descripción.- Aplicar el procedimiento de prueba "Acceso a los subsistemas", con el caso "Reportes de Error".

7.- Salir del sistema

Descripción.- Se deberá poder salir del sistema y desplegar la interfaz #1 "Login" desde las siguientes interfaces:

- #3 "Administrador de Proyectos Home"
- #6 "Documentos del Proyecto"
- #14 "Requerimientos de Cambio"
- #15 "Pruebas"
- #16 "Reportes de Error"

Cerciorarse de que no hay problema al volver a iniciar una sesión después de un logout.

Calidad y pruebas en el desarrollo de Sofware se terminó de imprimir en la Ciudad de México en septiembre de 2017. La producción editorial e impresión estuvo a cargo de Literatura y Alternativas en Servicios Editoriales S.C. Avenida Universidad 1815-c, Depto. 205, Colonia Oxtopulco, C. P. 04318, Delegación Coyoacán, Ciudad de México. RFC: LAS1008162Z1. En su composición se usaron tipos Minion Pro y Avenir. Se tiraron 100 ejemplares sobre papel.

Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Unidad Cuajimalpa