Universidad de las Ciencias Informáticas

Título: Estrategia de Pruebas Exploratorias para mejorar el rendimiento del Laboratorio Industrial de Pruebas de Software de Calisoft

Tesis en opción al título de
Máster en Calidad de Software

Autora:
Ing. Yeniset León Perdomo

Tutoras:
Dra. Ailyn Febles Estrada
Dra. Vivian Estrada Sentí

Ciudad de la Habana, septiembre de 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora del presente trabajo. Autorizo al Centro Nacional de Calidad de Software (Calisoft) de la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

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

_____

Ing. Yeniset León Perdomo

Autora

_____ _____

Dra. Ailyn Febles Estrada Dra. Vivian Estrada Sentí

Tutora Tutora

Dedicatoria

A mami, papi, mi hermano y a mi sobrina.

A mis abuelos, en especial a mamá Mulata.

A toda mi familia.

Agradecimientos

A mi familia toda, gracias por estar siempre pendiente de mí, le agradezco a Dios por haberme dado una familia tan especial.

A mis súper tutoras Ailyn y Vivian, gracias por exigirme tanto, son magníficas, las mejores del mundo. Gracias por creer que sí podía. Son un ejemplo a seguir.

A mi amiga y hermana Mary, gracias nena por ser tan especial y quererme tanto, gracias por permitirme tener un sobrino tan bello, son muy importante en mi vida.

A Pepe por ser amigo y hermano, nene sobran las palabras…

A Day por acompañarme en todos estos años y apoyarme, y a Liza también… en las buenas y en las malas chicas siempre estaré aquí.

A Tay, gracias amiga por siempre estar pendiente de mí y alentarme, gracias por hacerme ver que sí se puede.

A Sasha y Ramón gracias amigos por apoyarme, escucharme y siempre estar ahí para mí.

A todos mis compañeros del DPSW, los nuevos y los viejos, a todos. Chicos sin ustedes hubiese sido imposible, les agradezco cuanto tiempo y paciencia tuvieron conmigo y con las PE, jejeje. Son para mí una gran familia.

A Calisoft y sus integrantes, por haberme permitido todos estos años ser parte de una idea tan bella, todo lo que sé, se lo debo a ustedes…

A Yaimí por ser tan Yaimí, en verdad te agradezco por ayudarme tanto y presionarme para que pudiera terminar la tesis.

A Raquel Abella del CES de Uruguay, gracias por siempre responder mis correos…

A la UCI por ser mi segunda casa durante estos 10 años.

A todos mis amigos, ustedes saben quiénes son, los especiales, los que siempre han estado ahí en las buenas y en las malas. Gracias por su apoyo incondicional.

A todos aquellos que en algún momento me preguntaron ¿cómo va la tesis?...

RESUMEN

La calidad de software adquiere cada día mayor importancia en el mercado mundial, debido a las exigencias de los clientes y la competitividad entre las empresas. Cuba ha comenzado a formar parte de ese mercado debido a las perspectivas económicas que brinda. Es por ello que las empresas nacionales y principalmente la Universidad de las Ciencias Informáticas (UCI) se empeñan cada vez más en obtener productos con la mayor calidad posible para incluirse en la competencia.

Que el producto tenga calidad, implica que debe cumplir con ciertas exigencias entre las que están las requeridas por los clientes y la carencia de errores en el producto final. Las pruebas de software constituyen un elemento crítico para definir la calidad, éstas pueden ser vistas como parte del proceso de desarrollo de software o independiente del proceso seguido para su desarrollo. En éste último caso, que es el que se sigue en esta investigación, el proceso de prueba no tiene en cuenta la forma en que se lleva a cabo el desarrollo para definir las actividades a realizar.

El Centro Nacional de Calidad de Software (Calisoft) cuenta con un Laboratorio Industrial de Pruebas de Software (LIPS), laboratorio que se dedica a las pruebas de productos de software. La motivación para este trabajo surge de la necesidad de definir la estrategia de trabajo a utilizar para realizar pruebas exploratorias en el LIPS.

Con el fin de definir la estrategia, se realiza un estudio del arte en lo referente a la calidad y las pruebas de software. Esta información es organizada y resumida para la realización de las pruebas exploratorias de un producto. Se presentan las etapas definidas para la estrategia, las actividades, los artefactos y los roles. Se muestran las herramientas a utilizar y las métricas propuestas para obtener una evaluación del proceso y el producto. Se describe la aplicación de la estrategia definida en proyectos reales y se presentan las conclusiones y recomendaciones a partir de dicha experiencia.

Se concluye que la estrategia es una guía útil para realizar pruebas exploratorias de productos de software. Actualmente, la estrategia es utilizada en las pruebas de liberación del LIPS de Calisoft. Para cada proyecto de prueba, la estrategia es adaptada a las características del proyecto y al culminar el mismo, se evalúa la estrategia y se realizan las mejoras a partir de lo aprendido con la experiencia.

Palabras claves: calidad de software, pruebas de software, pruebas exploratorias, métricas.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICAS

INTRODUCCIÓN

En la última década el sector de desarrollo de productos y servicios de software se ha convertido en un impulsor dominante de la cada vez más creciente economía de la información (González 2007); es un sector con la capacidad de generar ventas millonarias, como por ejemplo, la India ha venido emergiendo como una gran potencia en la producción de software en los últimos tiempos. Ya en 2005, por conceptos de subcontratación de servicios y exportaciones de la Industria del Software (IndSW) y la información, el monto fue de 17 mil 200 millones de dólares, cifra a elevarse a 60 mil millones anuales para el 2010 (El Economista de Cuba 2009; Santos Hernández 2009).

Haciendo un análisis de los últimos de tres quinquenios hay un impacto positivo del "Standish Group" y sus "CHAOS Reports" en la caracterización del estado de la IndSW a través de la evaluación de sus proyectos exitosos, no satisfactorios y cancelados (Boston 2009). Hay una marcada tendencia, a largo plazo, a disminuir los proyectos cancelados aunque desde el 2002 con un 15% estos han aumentado en casi 10 puntos porcentuales; lo más significativo en los no satisfactorios es la estabilidad en un rango elevado (entre el 44% y el 53%) con una discreta disminución desde el 2004, mostrando que de manera general no se satisfacen los requisitos pactados con el cliente (ver anexo 1) (Domínguez 2009).

Aunque la IndSW en el mundo se ha desarrollado considerablemente en los últimos años, con un desarrollo acelerado, los resultados alcanzados no cubren las expectativas inicialmente vislumbradas debido básicamente a que la productividad que se alcanza en general, es baja, la cantidad de recursos a consumir (en tiempo principalmente) es alta y el trabajo realizado casi nunca tiene la calidad requerida. Los proyectos se concluyen en fecha posterior a lo planificado y los problemas no se detectan a tiempo en este medio indisciplinado y caótico de desarrollo, (SIME 1997; Álvarez. S 2000; Lage 2000).

Según el Pointe Technology Group. Inc. (Pointe Technology Group), los cinco problemas más comunes en el proceso de desarrollo de software son:

- Pobres requerimientos: si los requisitos no son claros, incompletos, demasiado general, o no comprobable, entonces hay grandes problemas.
- Cronogramas irrealistas: si el trabajo a desarrollar es mucho para tan poco tiempo, los problemas son inevitables.
- Inadecuadas pruebas: nadie sabrá si el programa es bueno hasta que el cliente se queja de un accidente o del propio sistema.

- Nuevas funcionalidades, características: se piden nuevas funcionalidades o características después que el desarrollo está en marcha, esto es extremadamente común.
- Falta de comunicación: si los desarrolladores no saben lo que se necesita, o el cliente tiene expectativas erróneas, los problemas están garantizados.

En Latinoamérica los problemas principales que existen en la IndSW, se han clasificado en cuatro categorías, (CaproSOFT 2003; BASTOS 2009):

- Déficit de recursos humanos en cantidad y calidad.
- Fortalecimiento de una institución como representante de los intereses de los productores de software.
- Subcontratación de servicios de software.
- La implementación de estándares de calidad de categoría internacional.

Como se muestra anteriormente uno de los factores que está presente dentro de los principales problemas de la IndSW es la calidad de éste, específicamente de los productos y servicios de software, identificado además como uno de los Factores Críticos de Éxito (FCE) para dicha industria.

Propiciar la calidad en el software es una actividad que ha surgido como consecuencia de la fuerte demanda y de la competencia, debido a la vorágine de ofertas en el mercado, convirtiéndose en una necesidad prioritaria para las organizaciones que lo desarrollan. El interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y exigentes, de cara a la solución de sus necesidades, elemento que marca el ritmo en la producción.

Desde el punto de vista del cliente y los usuarios, la calidad de un producto de software es percibida principalmente por las fallas que encuentran en el producto y por la gravedad que éstas tienen para el negocio del cliente. Para ser competitivas, las empresas desarrolladoras de software necesitan asegurarse de la calidad de sus productos previo a su instalación en el ambiente del cliente, (Pérez Lamancha 2006).

Uno de los aspectos más descuidados en este gran proceso de desarrollo es el de las pruebas, a pesar de ser una referencia obligada cuando se habla de mejorar la calidad. Éstas, aunque no proporcionan directamente calidad al sistema, si dan una visión clara de las debilidades observadas y de los riesgos asociados, sobre los cuales se puede trabajar para perfeccionar, catalogándose así como un técnica crucial dentro de la calidad y satisfacción del cliente, (Valdivia 2005).

Desde 1996 el país viene dando pasos para el ordenamiento de un trabajo continuo destinado a impulsar el uso y desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), lo cual condujo en enero de 2000 a la creación del Ministerio de la Informática y las Comunicaciones (MIC); los ingresos por comercio de TICs han mostrado un marcado aumento, principalmente por conceptos de servicios informáticos. Sin embargo, en cuanto a la producción de software se refiere, los ingresos han disminuido, evidenciándose en las producciones nacionales y de exportación en el período del 2008 al 2009, (AEC 2009). En el anexo 2 se muestran las tablas que representan los ingresos por comercio de TICs desde el 2007 hasta el 2009.

El país ha invertido millones de pesos (91.5 y 70.9 respectivamente) en equipos, construcción y montaje de tecnologías informáticas (AEC 2009), por lo tanto, es primordial para la Industria Cubana de Software (InCusoft) que aumente las producción de software y alcance un mayor auge en el mercado internacional, generando así mayores ingresos para contribuir al desarrollo de la economía. Para ello estos productos que son desarrollados tienen que tener la mejor y mayor calidad posible para que puedan ser comercializados, en un mundo donde se acrecientan las limitaciones económicas, tecnológicas y de comunicación.

Vinculado al desarrollo de la IncuSoft y con el objetivo de identificar los principales problemas que afectan el desempeño de la misma, se identificaron a través de revisiones bibliográficas los siguientes problemas (Febles Estrada 2003):

- Pobre planificación y organización del trabajo.
- Escaso desarrollo de proyectos de investigación y desarrollo.
- No se aplican estándares de calidad.
- Pobre la formación y la superación continua de los recursos humanos.
- Poca aplicación de técnicas de comercialización.
- Ausencia de una política salarial y de estimulación coherente para el personal informático.

La UCI es un centro docente-productivo, que gestiona el conocimiento y garantiza las soluciones informáticas tanto para el proceso de informatización de la sociedad cubana, como para las necesidades de diferentes clientes.

Ante el compromiso de satisfascer las múltiples y crecientes solicitudes de servicio que a ella llegan y basado en la premisa de garantizar la calidad al producto, es que se crea en el 2003 Calisoft, el cual es una organización enfocada a contribuir al desarrollo de la IncuSoft, facilitando la implementación de las mejores prácticas en el proceso de desarrollo y/o

mantenimiento de software. Es responsable de la verificación y validación de productos, procesos y organizaciones según normas nacionales e internacionales y de la asesoría, adiestramiento y formación continua de los especialistas en el país en los temas de calidad e ingeniería de software. Organización que puede comercializar los servicios de adiestramiento, consultoría y asesoría en estos temas, así como los servicios de pruebas y auditorías y promover el uso buenas prácticas para el desarrollo de software en la IncuSoft.

Este centro desde sus inicio contó con un laboratorio de pruebas, en el cual se liberaban todos los productos entregables de los proyectos nacionales y de exportación de la UCI. Debido al crecimiento de los artefactos a probar que venía experimentando el laboratorio y que no se contaba con una fuerza de trabajo permanente para darle respuesta a las solicitudes de pruebas realizadas, es que se crea en octubre del 2008 el LIPS. Tiene como misión principal lograr que todo producto elaborado en la UCI y en la InCuSoft, que se presente al laboratorio sea comprobado y evaluado según normas y estándares de calidad, antes de ser entregado al cliente, siendo esta evaluación confiable para los equipos de desarrollo y para los clientes.

En función de los cambios que se vienen implementeando en la UCI, basada en las buenas prácticas del trabajo de años anteriores y las experiencias adquiridas por los especialistas, se decide a finales del curso 2009 - 2010 crear el Departamento de Pruebas de Software (DPSW), al cual se le incorporó el LIPS como uno de sus grupos de trabajo, para lograr mayor organización, eficiencia y productividad en las pruebas que se le realizan a los software.

El funcionamiento del LIPS comienza desde que se realizada la solicitud de pruebas del producto hasta que se libera, [los artefactos son sometidos a diferentes períodos de pruebas (iteraciones)]. Previamente a la ejecución de la primera iteración, en algunas ocasiones, se le realiza una revisión al artefacto, nombradas Pruebas Exploratorias (PE). Esta prueba carece de una estrategia que la guíe, basándose solamente en la experiencia y conocimiento del especialista del laboratorio. No se tiene en cuenta la selección de la muestra del artefacto a probar, en la mayoría de los casos, debido a la magnitud del mismo, esto impide la retroalimentación rápida de la información necesaria para el proceso de pruebas venidero.

La ejecución empírica de las PE provoca un incremento gradual de gasto de recursos, dígase, tiempo, capital humano, tecnología y esfuerzo en las pruebas. Por tanto, se hace necesario identificar los problemas potenciales y encontrar los errores antes que inicie la

primera iteración. De esta manera las pruebas puedan planificarse, ejecutarse a tiempo y mitigar los impactos adversos para alcanzar los objetivos propuestos.

Desde hace algunos años, vinculado al desarrollo de la InCuSoft en la UCI, se viene desarrollando un grupo de investigaciones, unas de carácter docente y otras desde el punto de vista productivo, relacionadas con las áreas de procesos del desarrollo del software, las pruebas de software, la calidad en los procesos y en el producto final, cuyos resultados aporten métodos, estrategias, procedimientos y maneras que mejoren las prácticas en cuanto al modo de garantizar la calidad de cada producto UCI.

Para esta investigación y en coordinación con Calisoft, se le aplicó una encuesta a 270 personas perteneciente a 12 centros de desarrollo de la UCI (en el anexo 3 se muestra el diseño de la encuesta), la cual estuvo enfocada principalmente a las pruebas de software. Durante el procesamiento y análisis de los datos se han detectado diversos problemas, dentro de los que se destacan (en el anexo 4 se ejemplifican estos resultados a través de los datos cuantitativos obtenidos de la encuesta 1):

Planificación y seguimiento

Sí se planifican en el cronograma del proyecto las pruebas de software, sin embargo:

- No existe una adecuada planificación del trabajo, (en los cronogramas no se identifican qué tipo de prueba hacer en cada momento).
- No se ejecutan adecuadamente las pruebas en los proyectos.
- No se puede dar seguimiento al progreso del proyecto (cronogramas de pruebas, puntos de chequeo, etc.).
- Falta de objetividad en la apreciación de la evolución de las pruebas, al no poseer los conocimientos y mecanismos necesarios para detectar la magnitud del trabajo ha desarrollar.
- Los proyectos concluyen el desarrollo de sus artefactos, casi siempre, fuera de fecha, lo que reduce considerablemente el tiempo que se había planificado para la ejecución de las pruebas.

Calidad y productividad

- No se detectan con antelación los problemas en el software, afectando esto finalmente la calidad en los productos.
- Las pruebas se ejecutan principalmente al finalizar el desarrollo de los componentes, subsistema o producto.
- Desconocimiento de los elementos y las características que integran los conceptos de calidad y pruebas de software.

- Insuficiente aplicación de las pruebas de software en los proyectos.
- Pocos proyectos tienen definidos alguna estrategia de PE ó Testing Exploratorio.
- Déficit de recursos humanos en cantidad y calidad vinculados a las pruebas en los proyectos.

Organizativas

- Mala concepción y ejecución de los procedimientos, procesos o metodologías que guían las pruebas durante el desarrollo de los proyectos.
- Pobre formación y superación continua de los recursos humanos en temas de calidad.

De los problemas expuestos anteriormente se tratarán en esta investigación los que se muestran a continuación, puesto que son los que inciden directamente en el avance satisfactorio de los artefactos durante su liberación. Por lo tanto, se hace necesario aplicar técnicas para mejorar el rendimiento del LIPS, teniendo en cuenta que actualmente:

- No se puede obtener realimentación rápida de cierto producto o funcionalidad.
- No existe forma de conocer el producto rápidamente.
- No existe una forma de investigar y aislar un defecto en particular.
- No se puede obtener una evaluación de la calidad de las pruebas que se estén realizando.
- Se quiere evaluar la necesidad de diseñar pruebas para una porción del artefacto.
- No están organizadas y planificadas las PE que se hacían.
- En ocasiones se declaran fallidas o abortan las pruebas funcionales realizadas a un arterfacto a mediado de su ejecución, debido a que no se le realizaron pruebas anteriores para detectar los posibles errores que presentaba.
- Los artefactos generados por los proyectos llegan a la liberación por Calisoft con un gran número de errores.
- Se le realizan hasta más de cuatro iteraciones de pruebas a los artefactos.
- Se conoce el artefacto una vez que entra al LIPS.

Lo que provoca, entre otros, los siguientes conflictos:

- Los equipos de prueba son convocados innecesariamente.
- Duplicación de esfuerzos para llevar los artefactos a un estado anterior.
- Pérdida de tiempo y recursos en la ejecución de las pruebas.

Dada la necesidad de encontrar los principales errores en un breve plazo, identificar puntualmente los defectos encontrados, investigar las consecuencias que traen consigo un error en particular, con el fin de evaluar la necesidad de ejecutar otro tipo de pruebas en esa

zona y mitigar equivocaciones en el análisis de riesgo del producto, es que se rediseña la ejecución empírica de las PE en el LIPS, se le insertan nuevos conceptos como: Pruebas Exploratorias Iniciales (PEI) y se define el siguiente **problema de investigación:**

¿Cómo ser más eficiente durante la ejecución de las pruebas de software en el Laboratorio Industrial de Pruebas de Software?

En esta investigación se trata la **eficiencia** mediante las variables: tiempo, rendimiento y eficacia; desglosadas en: la medición del esfuerzo que se requiere para alcanzar los objetivos. La capacidad de alcanzar las metas programadas con el mínimo de recursos disponibles (uso adecuado de factores materiales y humanos) y tiempo de ejecución de las pruebas, logrando su optimización y cumpliendo con la calidad propuesta. En el anexo 5 se presenta un análisis de los indicadores y sub-indicadores asociados a la variable eficiencia.

Se formuló la siguiente **hipótesis de investigación:**

Con la implementación de una estrategia de pruebas exploratorias en el Laboratorio Industrial de Pruebas de Software, se garantiza una mayor eficiencia durante la ejecución de las pruebas.

Para enfrentar este problema, se definió como **objeto de investigación**: el proceso del control de la calidad de productos de software y como **campo de acción:** las pruebas de liberación en el Laboratorio Industrial de Pruebas de Software.

Para responder al problema de investigación, se definió el **objetivo general**:

Elaborar una estrategia de pruebas exploratorias para mejorar el rendimiento del Laboratorio Industrial de Pruebas de Software.

A partir del análisis del objetivo general se derivan los siguientes **objetivos específicos**:

- Estudiar el marco teórico sobre el estado actual de las pruebas de software en el mundo y principalmente en la UCI.
- Determinar los principales elementos a incluir en la estrategia de pruebas exploratorias.
- Elaborar la estrategia de pruebas exploratorias para mejorar el rendimiento del LIPS.
- Validar la estrategia definida para las pruebas exploratorias en proyectos reales.

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones (Hernández León 2002).

En la investigación como **métodos científico de investigación** utilizados se destacan los siguientes:

Métodos teóricos: *Método hipotético-deductivo* para la elaboración de la hipótesis central de la investigación y plantearse objetivos específicos a partir del problema concreto, para proponer nuevas líneas de trabajo a partir de los resultados parciales. *Método sistémico* para lograr que los elementos que forman parte de la estrategia sean un todo que funcione de manera armónica. *Método histórico-lógico y el dialéctico* para el estudio crítico de los trabajos anteriores, para utilizar éstos como punto de referencia y comparación de los resultados alcanzados. El estudio comparativo sirvió para establecer similitudes y diferencias en las distintas estrategias de pruebas presentadas en el capítulo I.

Métodos lógicos: *Método analítico-sintético* al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta. Método de *idealización-modelación* para explicar por qué la estrategia de PE seleccionada es la que más se ajusta a las características del LIPS.

Métodos empíricos: *Método experimental* para comprobar la utilidad de los resultados obtenidos a partir de la estrategia definida. *Método de la entrevista y la encuesta* fueron vitales para el diagnóstico de la organización relacionado con las pruebas de software. Para establecer los elementos necesarios para la lógica de la estrategia, avalar los conceptos que se manejan en la investigación, medir el alcance y la importancia que tiene la temática. Captar la información cualitativa y cuantitativa del fenómeno, conocer los criterios sobre la forma en que se organiza y se lleva a cabo la calidad y pruebas de software en la UCI, así como las posibles soluciones que se proponen en la investigación. Para ello se entrevistaron y encuestaron personas involucradas en la producción de software en la UCI y especialistas del laboratorio de prueba del DPSW.

Métodos matemáticos: Para obtener los atributos a evaluar en las métricas a incluir en la estrategia*. Los métodos de experto* para la definición de los aspectos a incorporar en la lista de prioridades de la selección de la muestra. *Métodos estadísticos* para el procesamiento de las encuestas y la definición de las métricas.

La **Significación práctica** del trabajo es la siguiente:
La definición y fundamentación de una estrategia que sirve como guía para realizar pruebas exploratorias de un producto de software en el LIPS. La utilización de esta estrategia hará más fácil, cómoda y organizada la labor de los especialistas de calidad durante la liberación

de los artefactos, garantizando alcanzar más rápidamente una disciplina de trabajo. Disminuirá el tiempo de ejecución de las iteraciones de prueba y optimizará los recursos. Estos resultados tendrán amplia aplicación en la UCI y en particular en el LIPS. La estrategia puede ser adaptada a otros centros u organizaciones.

Estructura de los capítulos

La tesis quedó estructura en tres capítulos:

El capítulo I, referido al marco teórico y referencial de la investigación, donde se brinda una visión general de las pruebas de software, definiendo los principales términos que serán usados en esta tesis, los distintos tipos de pruebas y las técnicas existentes.

El Capítulo II, donde se presenta la estrategia de PE, estrategia propuesta para evaluar los artefactos que entran a liberarse al LIPS. Se presentan las principales características de la estrategia de prueba, brindando una visión global de sus etapas, actividades, artefactos y roles. Se muestran un grupo de métricas internas y externas, que permitirán medir el proceso y el producto y finalmente se introduce el uso de herramienta automatizada para la gestión de las NC.

El Capítulo III, describe la aplicación práctica de la estrategia y se valida la propuesta. Se detallan las conclusiones de esta tesis y los posibles trabajos a futuro.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Introducción

En el Capítulo I se presentan los principales conceptos relacionados con la Calidad de Software. Se definen prueba de software, verificación, validación y su relación con las pruebas. Se realiza un breve estudio sobre el estado actual de las pruebas de software en el mundo, específicamente en la UCI y algunas organizaciones que las aplican como uno de sus servicios fundamentales (laboratorios de pruebas de software). Finalmente se muestran las distintas definiciones sobre pruebas exploratorias y se precisan los conceptos que se definieron para este trabajo. El esquema de la Figura 1 constituye el "hilo conductor" de los aspectos fundamentales del objeto de estudio teórico.

Figura 1: Esquema de la investigación del capítulo I.

1.1 Calidad de Software

La calidad de software es un problema actual que afecta tanto a los productores de software como a los clientes. Con el aumento de la informatización a escala mundial la demanda de software crece exponencialmente y los desarrolladores le han brindado poco interés a la calidad de sus productos. Sucede que muchas veces los clientes reciben el software habiéndose violado la etapa de pruebas. El interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar productos poco fiables o que realmente no dan respuesta a sus necesidades. Ahora bien, ¿qué es la calidad del software?

Según (Pressman 2005), la calidad del software es la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente.

Para (Lovelle 1999) la calidad es el conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas. La ausencia de defectos, la aptitud para el uso, la seguridad, la confiabilidad y la reunión de especificaciones son elementos que están involucrados en este concepto.

En este sentido la autora coincide con Pressman en que la calidad del software debe ser especificada, planificada, administrada, medida y certificada. Esto implica una visión integral que arroja la comprobación del software, con el fin de lograr un mayor grado de satisfacción y confianza del cliente hacia la organización productora de software. Constituye entonces las pruebas del software, tarea de alta prioridad para las empresas productoras, (Pressman 2002).

La satisfacción del cliente es algo determinante aunque pueda parecer poco significativo. El cliente casi siempre tiene razón y es el que ubica el producto en los primeros puestos de la tabla de calidad por aclamación de los usuarios (Fernández 2001).

A la hora de definir la calidad del software se debe diferenciar entre la calidad del producto de software y la calidad del proceso de desarrollo. No obstante, las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

La calidad del producto de software se diferencia de la calidad de otros productos de fabricación industrial, puesto que el software tiene ciertas características especiales, (Scalone 2006):

1. El software es un producto mental, no restringido por las leyes de la física o por los límites de los procesos de fabricación. Es algo abstracto y su calidad también lo es.
2. Se desarrolla, no se fabrica. El coste está fundamentalmente en el proceso de diseño, no en la producción. Y los errores se introducen también en el diseño, no en la producción.
3. El software no se deteriora con el tiempo. No es susceptible a los efectos del entorno y su curva de fallos es muy diferente a la del hardware. Todos los problemas que surjan durante el mantenimiento estaban desde el principio y afectan a todas las copias del mismo; no se generan nuevos errores.
4. Es artesanal en gran medida. El software, en su mayoría, se construye a medida, en vez de ser construido ensamblando componentes existentes y ya probados, lo que

dificulta aún más el control de su calidad. Aunque se ha escrito mucho sobre la reutilización del software, hasta ahora se han conseguido pocos éxitos tangibles.

5. El mantenimiento del software es mucho más complejo que el mantenimiento del hardware. Cuando un componente de hardware se deteriora se sustituye por una pieza de repuesto, pero cada fallo en el software implica un error en el diseño o en el proceso mediante el cual se tradujo el diseño en código de máquina ejecutable.
6. Es engañosamente fácil realizar cambios sobre un software, pero los efectos de estos cambios se pueden propagar de forma explosiva e incontrolada.
7. Como disciplina, el desarrollo de software es aún muy joven, por lo que las técnicas de las que disponemos aún no son totalmente efectivas o no están totalmente calibradas.
8. El software con errores no se rechaza. Se asume que es inevitable que el software presente errores.

Es importante destacar que la calidad del software debe ser considerada en todos sus estados de evolución (especificaciones, diseño, código, etc). No basta con tener en cuenta la calidad del producto una vez finalizado, cuando los problemas de mala calidad ya no tienen solución o la solución es muy costosa. La problemática general a la que se enfrenta el software es:

1. Aumento constante del tamaño y complejidad de los programas.
2. Carácter dinámico e iterativo a lo largo de su ciclo de vida, es decir que los programas de software a lo largo de su vida cambian o evolucionan de una versión a otra para mejorar las prestaciones con respecto a las anteriores.
3. Dificultad de conseguir productos totalmente depurados, ya que en ningún caso un programa será perfecto.
4. Se dedican elevados recursos monetarios a su mantenimiento, debido a la dificultad que los proyectos de software entrañan y a la no normalización a la hora de realizar los proyectos.
5. No suelen estar terminados en los plazos previstos, ni con los costes estipulados, ni cumpliendo los niveles deseables de los requisitos especificados por el usuario.
6. Incrementos constantes de los costes de desarrollo debido entre otros, a los bajos niveles de productividad.
7. Los clientes tienen una alta dependencia de sus proveedores por ser en muchos casos aplicaciones a "medida".
8. Procesos artesanales de producción con escasez de herramientas.
9. Insuficientes procedimientos normalizados para estipular y evaluar la calidad, costes y productividad.

En la introducción de esta investigación se hizo referencia a los cinco problemas más comunes en el proceso de desarrollo de software, cúales serán entonces las cinco soluciones a estos problemas. El Pointe Technology Group. Inc. (Pointe Technology Group) las clasifica de la siguiente manera:

- Sólidos requerimientos: claros, completos, detallados, coherentes, accesibles, verificables requisitos que se acordó por todos los implicados. El uso de prototipos para ayudar a concretar los requisitos.
- Cronogramas realistas: tiempo suficiente para la planificación, diseño, pruebas, corrección de errores, repetir las pruebas, los cambios y la documentación, el personal debe ser capaz de completar el proyecto sin “quemarse”.
- Adecuadas pruebas: iniciar la prueba desde el principio, repetición de la prueba después de correcciones o cambios, incorporar en el cronograma de desarrollo el tiempo adecuado para la prueba y corrección de errores.
- Adherirse a los requisitos iniciales tanto como sea posible: estar preparados para defenderse de los cambios y adiciones, una vez comenzado el desarrollo y estar preparado sobre todo para explicar las consecuencias que traerían los cambios. Si los cambios son necesarios, deben ser adecuadamente reflejados en el cronograma de desarrollo. Si es posible, utilizar el prototipado rápido durante la fase de diseño para que los clientes pueden ver lo que se va cambiando. Esto les proporcionará un mayor nivel de confort con los requisitos que ellos seleccionaron y la posibilidad de minimizar los cambios más adelante.
- Comunicación: requieren recorridos e inspecciones cuando sea necesario, hacer uso extensivo de herramientas de comunicación de grupo - correo electrónico, herramientas de seguimiento de fallos de red y herramientas de gestión del cambio, las capacidades de la intranet, etc.; asegurar que la documentación esté disponible y actualizada - preferiblemente en formato electrónicos, no de papel; promover el trabajo en equipo y la cooperación; uso temprano de prototipos en lo que las expectativas de los clientes se aclaran.

La calidad del software debe implementarse en todo el ciclo de vida del software. Las distintas actividades para la implantación del control de calidad en el desarrollo de software son: (1) aplicación de metodología y técnicas de desarrollo, (2) reutilización de procesos de revisión formales, (3) prueba del software, (4) ajustes a los estándares de desarrollo, (5) control de cambios, mediciones y recopilación de información; y (6) gestión de informes sobre el control de calidad, (Pressman 2002).

La calidad se logra principalmente a través de la gestión de la calidad, la cual, según ISO 9000:2000, consiste en la realización de actividades coordinadas que permiten dirigir y controlar una organización en lo relativo a la calidad, (ISO 9000:2000).

1.1.1. Gestión y control de la calidad del software

La gestión de la calidad de software es una actividad esencial en cualquier empresa de software para asegurar la calidad de sus productos y la competitividad frente a la oferta del mercado. Es un conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades. (ISO 9000:2000) se basa en la determinación y aplicación de las políticas de calidad de la empresa (objetivos y directrices generales). La gestión o administración de la calidad se aplica normalmente a nivel empresa. También puede haber una gestión de la calidad dentro de la gestión de cada proyecto.

Desde el punto de vista de la calidad, la gestión de la calidad del software está formada por cuatro partes, las cuales son: (1) planificación de la calidad del software, (2) control de la calidad del software, (3) aseguramiento de la calidad del software y (4) mejora de la calidad del software.

1.1.2. Control de la calidad del software

Según la Norma ISO (ISO 9000:2000), el control de la calidad es la parte de la gestión de la calidad orientada al cumplimiento de los requisitos de la calidad.

El control de la calidad del software son las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales: (1) mantener bajo control un proceso y (2) eliminar las causas de los defectos en las diferentes fases del ciclo de vida, (Pressman 2002). Está formado por actividades que permiten evaluar la calidad de los productos de software desarrollados. El aspecto a considerar en el control de la calidad del software es la “Prueba de Software”.

1.2. Pruebas de Software

1.2.1. Organizaciones que realizan pruebas de software en el mundo

La innovación tecnológica, velocidad en el desarrollo y la satisfacción del cliente se ha convertido en la consigna de las organizaciones que quieren sobrevivir y cada vez ser más competitivas en el mundo. Es esta una de las razones por las que el desarrollo de la IndSW ha pasado a ser prioridad de los gobiernos en muchos países de la región.

En la actualidad, mientras que los servicios de software están creciendo a un promedio de alrededor de 10-12% a nivel mundial, las pruebas están creciendo a más del 50 % cada año.

La India, nombrado unos de las tres “I” en el desarrollo del software y uno de los países con mayor éxito en esta industria, se ha convertido en un líder en el mercado de pruebas de software con un número creciente de empresas de desarrollo de software.

"La combinación de costos, la comunicación, la exposición a diversos ámbitos, ensayos y prueba de herramientas le proporciona una clara ventaja a la India en pruebas de software", según (Hexaware Technologies). La oportunidad de mercado para las empresas indias de pruebas en alrededor de $ 8 millones a finales del 2008, comparados con los $ 2-3 mil millones del 2007.

En el anexo 6 se presentan y describen algunos de los laboratorios de prueba de software existentes o en construcción en Latinoamérica y demás regiones del mundo, según estudios realizados por la autora. Se puede concluir que estos laboratorios tienen aspectos en común, todos expresan que es esencial para su desarrollo:

- Establecer una estrategia de pruebas.
- Definir sus procesos, infraestructura, herramientas (automatizadas fundamentalmente) y técnicas.
- Establecer la gobernabilidad, las métricas y el marco de la comunicación.
- Realizar el plan de gestión del conocimiento para atender las demandas de recursos y al modelo de prestación de configuración centralizada.
- Desarrollar los recursos humanos. Contar con un equipo multidisciplinario para el diseño y ejecución de las pruebas.

En Cuba, hasta la creación de Calisoft, no existía ninguna organización que se dedicara por completo a los temas relacionados con la calidad y pruebas de software. Esto era algo que las propias empresas productoras de software debían garantizar. Con la creación de este centro se garantiza este servicio, el cual actualmente su proyección no es solo para los proyectos UCI sino para toda la InCuSoft.

1.2.2. Pruebas de software en la UCI. LIPS

Calisoft desde sus inicios en el 2003 ha experimentado diversos cambios estructurales, organizativos y un marcado aumento de sus recursos humanos, debido al incremento de los proyectos productivos de la universidad, lo que proporcionó la ampliación y perfeccionamiento en los servicios que brinda. En la Figura 2 se muestra la estructura organizativa actual de Calisoft, distinguiéndose por los cambios estructurales que han ocurrido desde que fue creado en el año 2003.

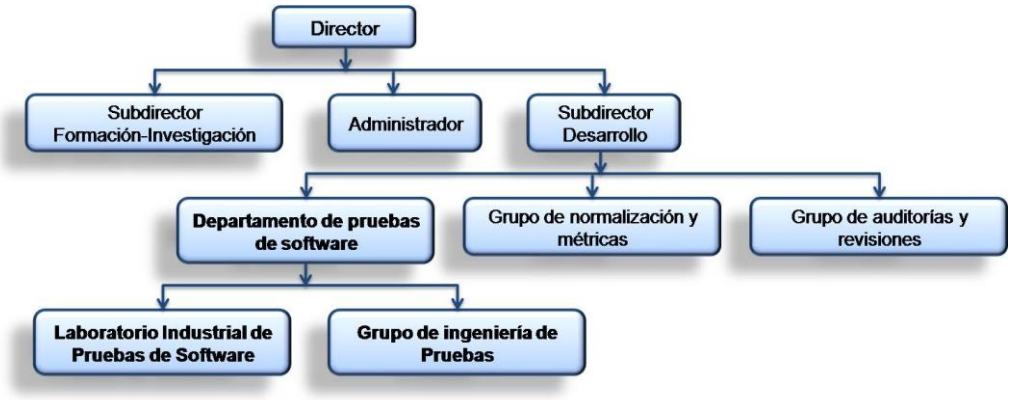


Figura 2: Estructura organizativa de Calisoft.

Como se muestra en la Figura 2 este centro cuenta con el DPSW. Este departamento ha venido evolucionando desde el 2003, año de su creación, cuando se le conocía como laboratorio de calidad y se contaba para realizar las pruebas solamente con cuatro especialistas y un grupo de diez estudiantes de tercer año de la carrera.

Actualmente el DPSW se compone de dos grupos: el LIPS, enfocado en la evaluación de productos desde el punto de vista funcional y el Grupo de Ingeniería de Pruebas (GIPS) donde se realizan otros tipos de pruebas (carga, estrés, rendimiento, etc.) y se analizan, planifican y preparan las pruebas por los especialistas del mismo.

El LIPS cuenta con dos laboratorios, con una capacidad de 30 puestos de trabajo a tiempo completo para las pruebas y como fuerza de trabajo, todos los estudiantes de 2do año de la UCI (2000 estudiantes aproximadamente). Cuenta además con 35 especialistas de calidad de software encargados de dirigir todo el proceso de pruebas en el laboratorio.

Como se muestra no sólo la estructura ha sido la que ha evolucionado en el tiempo, sino también ha aumentado el número de especialistas, la preparación de éstos es muy superior, los procesos y procedimientos están definidos de forma más coherente y adaptados a los nuevos requerimientos de la producción en la UCI. Además se han ampliado los tipos de prueba que se realizan, de forma que se garantizan más atributos de calidad en los productos y las herramientas que se utilizan ya no son todas manuales, sino que se han introducido herramientas automatizadas en el proceso.

Todo esto ha propiciado una mayor seriedad y cultura de calidad en los proyectos, lo cual se evidencia a partir de la obtención de resultados positivos en las pruebas de aceptación con el cliente de los productos que han sido liberados.

1.2.3. Pruebas de software. Definiciones

La incorporación de las prácticas de calidad en el desarrollo de software tiene numerosos beneficios. En cuanto al cliente, se consigue una mejora de su satisfacción, de la fiabilidad del software, se reducen los errores en explotación y se logra el cumplimiento de los requisitos. Desde la perspectiva de una organización, se consigue verificar que se han implantado las características que indicó el usuario, asegurar que los procesos se aplican de la forma adecuada y que esos procesos se mejoran con el tiempo, (Brualla 2005).

La primera referencia a las pruebas de software puede ser rastreada a 1950, pero fue recién en 1957 que la prueba fue distinguida del debugging, (Hetzel 1988). Dijkstra en 1970 presentaba una importante limitación, que "la prueba de software puede ser usada para mostrar la presencia de bugs, pero nunca su ausencia", (Dijkstra 1970).

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

Existen enfoques dinámicos y estáticos para las pruebas. Los enfoques dinámicos apuntan a ejecutar una parte o todo el software para determinar si funciona según lo esperado. El enfoque estático se refiere a la evaluación del software sin ejecutarlo usando mecanismos automatizados (herramientas asistidas) y manuales tales como controles de escritorio, inspecciones y revisiones, (Daich 1994).

En este trabajo se adopta el enfoque dinámico de las pruebas, tomando como definición de prueba la de SWEBOK, (SWEBOK 2004). En SWEBOK se definen prueba, prueba de software, verificación dinámica y comportamiento esperado de la siguiente manera:

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

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

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

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

En la literatura existen otras definiciones. Por ejemplo en (IEEE-610-12:1990) de ANSI / IEEE, 1990 se define la prueba: "Es el proceso de operar un sistema o componente bajo condiciones específicas, observar o registrar los resultados, y realizar una evaluación de algún aspecto del sistema ocomponente".

Los principios básicos de las pruebas de software son: (1) a todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos del cliente; (2) las pruebas deberían planificarse mucho antes que empiecen; (3) las pruebas deberían empezar por "lo pequeño" y progresar hacia "lo grande"; y (4) para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente.

1.2.3.1. Verificación, validación y prueba

Los términos verificación y validación suelen usarse indistintamente en algunos contextos. Para las pruebas del software ambos términos indican conceptos diferentes:

La definición dada por (IEEE-610-12:1990) de ANSI / IEEE, 1990 de verificación y validación es la siguiente:

- Verificación: proceso de evaluación de un sistema o componente para determinar si un producto de una determinada fase de desarrollo satisface las condiciones impuestas al inicio de la fase.
- Validación: proceso de evaluación de un sistema o componente durante o al final del proceso de desarrollo para determinar cuando se satisfacen los requerimientos especificados.

Boehm, (Boehm 1984) usa dos preguntas para clarificar la diferencia entre verificación y validación:

- Verificación: ¿Estamos elaborando correctamente el producto?
- Validación: ¿Estamos elaborando el producto correcto?

CMMI, (CMMI 2002) define el propósito de la verificación y la validación de la siguiente manera:

- El propósito de la verificación es: asegurar que los productos internos seleccionados cumplen con su especificación de requerimientos. Los métodos de verificación pueden

ser, entre otros: inspecciones, revisiones por pares, auditorias, recorridas, análisis, simulaciones, pruebas y demostraciones.

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

Según Kit (Kit 1995), la verificación es el proceso de evaluar, revisar, inspeccionar y hacer controles de escritorio de productos intermedios, tales como especificaciones de requerimientos, especificaciones de diseño y código. En el caso del código se refiere a un análisis estático del mismo y no de su ejecución dinámica. La validación implica ejecutar el código. La prueba la define como la verificación más la validación.

En general en la bibliografía, la prueba es tomada como una parte del proceso de verificación y validación y bajo el entendido de que la prueba requiere ejecutar el código. En este trabajo se utiliza el enfoque de SWEBOK, (SWEBOK 2004), donde se define que:

La salida observada de la ejecución del programa puede ser comparada mediante:
Prueba para la verificación: El comportamiento observado es revisado contra las especificaciones.

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

1.2.3.2. Elementos de la prueba

A continuación se definen los principales conceptos usados para la prueba de un producto de software:
Una prueba (test) es:

- Una actividad en la que un sistema o componente es ejecutado bajo condiciones especificadas, los resultados son observados o registrados y una evaluación es hecha de algún aspecto del sistema o componente.
- Un conjunto de uno o más casos de prueba, (IEEE-Std-610-12:2002).

Un caso de prueba (test case) (CP) es un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y poscondiciones de ejecución, desarrollados con un objetivo particular o condición de prueba, tal como ejercitar un camino de un programa particular o para verificar que se cumple un requerimiento específico, (IEEE-Std-610-12:2002).

El “caso de prueba bueno” es aquel que tiene alta probabilidad de detectar un defecto aún no descubierto. El “caso de prueba exitoso” es aquel que detecta un defecto aún no descubierto, (Pérez Lamancha 2006).

En el caso de esta investigación los CP son diseñados por el equipo de desarrollo y revisados y probados por los especialistas del DPSW, comprobando que recorren todos los caminos posibles del software, todas las funcionalidades y que cumplen con los requerimientos pactados inicialmente con el cliente.

Un procedimiento de prueba (test procedure) es:

- Instrucciones detalladas para la configuración, ejecución y evaluación de los resultados para un caso de prueba determinado.
- Un caso de prueba puede ser usado en más de un procedimiento de prueba, (IEEE-Std-610-12:2002).

Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar acabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos y cuánto esfuerzo, tiempo y recursos se van a requerir. Cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas; y la agrupación y evaluación de los datos resultantes, (Scalone 2006).

Las características generales de las estrategias de prueba de software son las siguientes:

1. La prueba comienza en el nivel módulo y trabaja “hacia fuera”, hacia la integración de todo el sistema basado en computadora.
2. Diferentes técnicas de prueba son apropiadas en diferentes momentos.
3. La prueba la realiza el que desarrolla el software y un grupo de prueba independiente.
4. La prueba y la depuración son actividades, pero la depuración se puede incluir en cualquier estrategia de prueba.

Un resultado real (actual result) es el comportamiento producido u observado cuando un componente o sistema es probado, (ISTQB. 2005).

Un resultado esperado (expected result) es el comportamiento predecido por la especificación u otra fuente, del componente o sistema a ser probado bajo condiciones especificadas, (ISTQB. 2005).

Un ciclo de prueba (test cycle) es la ejecución del proceso de prueba contra una versión identificada del producto a probar, (ISTQB. 2005).

Los datos de prueba (test data) son los datos que existen (por ejemplo, en una base de datos) antes de que una prueba sea ejecutada y que afecta o es afectado por el componente o sistema a probar, (ISTQB. 2005).

La ejecución de la prueba (test execution) es el proceso de ejecutar una prueba para el componente o sistema a probar, produciendo el resultado real, (ISTQB. 2005).

1.2.4. Consideraciones respecto a la prueba

El problema fundamental respecto a la prueba de software es que no se puede probar completamente un programa, por lo que antes de comenzar las pruebas se deben revisar todos los casos de prueba. Otro punto importante a tener en cuenta es la actitud que debe tener la persona que realiza las pruebas.

La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

1.2.4.1. La necesidad de probar

La prueba de software es un elemento crítico e imprescindible para la garantía de la calidad, y de ahí la necesidad de aplicarla. A través de la prueba se ve la medida en que las funcionalidades del software se corresponden con las especificaciones establecidas y los datos que va arrojando constituyen un indicador de la fiabilidad del mismo.

Dentro de las causas que fundamentan las pruebas, las cuatro más sobresalientes son:
Propensión a equivocarse: El ser humano es propenso a cometer equivocaciones; éstas se manifiestan en diversos problemas contenidos en los modelos (defectos o faltas) y pueden manifestarse como fallas en tiempo de ejecución.

Fallas de hardware: La infraestructura empleada para el desarrollo de software (hardware, sistemas operativos, compiladores, etc.) no está exenta de fallas, lo que introduce defectos adicionales o permite que subsistan inadvertidos los que introdujo el desarrollador.

Creatividad del desarrollado: El desarrollo de software es una labor creativa y por ello es común que el producto desarrollado no coincida con el modelo contenido en las especificaciones, (Fernández 2002).

Costo de corrección de errores: se multiplica a medida que avanza el desarrollo (ver Tabla 1) (Dustin 2008), por lo general en los proyectos UCI esto no se tiene en cuenta de forma cuantitativa, se evidencia cuando el equipo de desarrollo del proyecto se encuentra

presionado por la necesidad de cumplir con las fechas establecidas en el cronograma y el proceso de pruebas no se cumple o se ejecuta de una manera desorganizada, sin método y sin considerar los tiempos establecidos para esta fase. El resultado es un software sin las pruebas mínimas requeridas y sin el nivel de calidad esperado.

Tabla 1: Aumento del costo de corrección de errores a medida que avanza el desarrollo.

Fase	Costo
Definición	$1
Alto nivel de diseño	$2
Bajo nivel de diseño	$5
Código	$10
Prueba de unidad	$15
Prueba de integración	$22
Prueba de sistema	$50
Después de la entrega	$100

1.2.4.2. No es posible probar completamente un programa

Para probar completamente un sistema se deben ejercitar todos los caminos posibles del programa aprobar. Myers mostró en 1979 un programa que contenía un loop y unos pocos if, este programa tenía 100 trillones de caminos, un probador podía probarlos todos en un billón de años.

La prueba exhaustiva requiere probar el comportamiento de un programa para todas las combinaciones validas e invalidas de entradas. Además se debe probar esto en cada punto donde se ingresan datos, bajo cada estado posible del programa en ese punto. Esto no es posible, ya que el tiempo requerido es prohibitivo.

El objetivo debe ser maximizar la producción de las pruebas, esto es, maximizar el número de los errores encontrados por un número finito de los casos de prueba, (Myers G. 2004).

1.2.4.3. Actitud frente a las pruebas

La prueba debe ser visto como el proceso destructivo de encontrar errores (cuya presencia se asume) en un programa. Si el propósito del probador es verificar que el programa funciona correctamente, entonces el probador esta fallando al conseguir su propósito cuando encuentra defectos en el programa. En cambio, si el probador piensa que su tarea es encontrar fallas, los buscará con mayor ahínco que si piensa que su tarea es verificar que el programa no las tiene, (Pérez Lamancha 2006).

El probador debe adoptar una actitud destructiva hacia el programa, debe querer que falle, debe esperar que falle y debe concentrarse en encontrar casos de prueba que muestren sus fallas, (Kaner C. 1999).

1.2.5. Clasificaciones de las pruebas

Existe en la bibliografía distintas formas de clasificar los tipos de prueba existentes. En esta investigación los tipos de prueba se describen en función de las características de calidad definidas en la norma ISO 9126, (NC-ISO-IEC-9126-1:2005) y teniendo en cuenta las sub-características que se plantean por cada una, las pruebas se ejecutarán chequeando los atributos de McCall, (McCall 1977).

En el anexo 7 se muestra la Tabla 16 donde se precisan los tipos de prueba que están definidos en el DPSW por cada característica de calidad, aunque se sigue profundizando en cada una, para particularizarlas y especializarlas, teniendo en cuenta el tipo de aplicación que debe probarse. Además se estudian con mayor detalle cada una de las características, de manera que se determinen otras pruebas que deban realizarse, para garantizar el cumplimiento de cada una en el artefacto que se prueba, lo que aumentará el rigor de las pruebas y como consecuencia, la calidad de los productos.

Además de estos tipos de pruebas, todas realizadas por especialistas del DPSW en su nivel más complejo, se revisa de forma general en los proyectos la documentación generada durante el desarrollo y que forma parte de los entregables comprometidos con el cliente. A esta revisión se le llama evaluación estática, definida como: las técnicas de evaluación estática de artefactos del desarrollo se les conoce de modo genérico por revisiones. Las revisiones pretenden detectar manualmente defectos en cualquier producto del desarrollo. Manualmente quiere decir, que el producto en cuestión (sea requisito, diseño, código, etc.) es analizado mediante la lectura del mismo, sin ejecutarlo.

1.2.6. Otros tipos de pruebas

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

1.2.6.1. Pruebas de regresión

Su objetivo es verificar que no ocurrió una regresión en la calidad del producto luego de un cambio, asegurándose que los cambios no introducen un comportamiento no deseado u errores adicionales. Implican la reejecución de alguna o todas las pruebas realizadas anteriormente (Black 2002).

Después que los probadores reportan los defectos, los desarrolladores generan una nueva versión del software, en la cual los defectos supuestamente han sido removidos. La cuestión de fondo es: ¿Cuánta prueba de la versión n es necesario hacer usando las pruebas ejecutadas contra la versión n-1? (Whittaker 2000).

Cualquier arreglo de un problema detectado puede (Whittaker 2000):

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

1.2.6.2. Prueba de humo

Las pruebas de humo son un conjunto de pruebas aplicadas a cada nueva versión, su objetivo es validar que las funcionalidades básicas de la versión se cumplen según lo especificado. Estas pruebas buscan grandes inestabilidades o elementos clave faltantes o defectuosos, que hacen imposible realizar la prueba como fue planificado para la versión. Si la versión no pasa las pruebas de humo, no se comienza la ejecución de las pruebas planificadas de la versión, (Kaner C. 2001).

1.2.6.3. Herramientas automatizadas

Hay herramientas que apoyan diversos aspectos de la prueba, tanto para la ejecución de las pruebas, la virtualización de los laboratorios de pruebas, como para el control y seguimiento de las No Conformidades (NC). En el anexo 8 se presentan las herramientas que son utilizadas por el DPSW con una breve descripción de las mismas y el momento en que se ejecutan (ver Tabla 17). Además en este anexo se presenta la Tabla 18 donde se listan las herramientas que fueron estudiadas para esta investigación con sus principales características.

1.2.7. Técnicas de prueba

En la literatura existen distintas formas de agrupar las técnicas de prueba. En esta investigación se agrupan fusionando la clasificación usada en (SWEBOK 2004) y (Kaner C. 2001). En el anexo 9 en la Tabla 19 se muestran las agrupaciones definidas y las técnicas comprendidas en cada una, enfantizando en la Técnica de Caja Negra, la cual se aplica en el LIPS. Además se presenta una breve explicación de cada técnica.

1.2.7.1. Técnica de caja negra y técnicas basadas en la experiencia

La estrategia definida en este trabajo es para realizar pruebas exploratorias perteneciente a la técnicas basada en la experiencia y dentro de ella fundamentalmente se realizarán prueba funcional de productos de software.

Se llama técnicas de caja negra o técnicas de prueba funcional a aquellas donde los casos de prueba se derivan a partir de la especificación del sistema.

Cuando se considera el software de computadora, la prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Este tipo de prueba examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.

Los métodos de prueba de la caja negra se centran en los requisitos funcionales de software. La prueba de la caja negra permite al ingeniero del software obtener conjuntos de condicionales de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de la caja negra no es una alternativa a las técnicas de prueba de la caja blanca. La prueba de la caja negra intenta encontrar errores de las siguientes categorías: 1) funciones incorrectas o ausentes; 2) errores de interfaz; 3) errores en estructuras de datos o en acceso a bases de datos externas; 4) errores de rendimiento y 5) errores de inicialización y terminación, (Scalone 2006).

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

1.2.7.1.1. Pruebas exploratorias

El término “pruebas exploratorias” fue introducido por Cem Kaner, se refiere a ejecutar las pruebas a medida que se piensa en ellas, sin gastar demasiado tiempo en preparar o explicar las pruebas confiando en los instintos, (Pérez 2007).

James Bach define como pruebas exploratorias al proceso simultáneo de exploración del producto (aprendizaje), diseño y ejecución de pruebas, (Bach 2002).

Las mismas constituyen una técnica de prueba en la cual el probador controla activamente el diseño mientras son realizadas y utiliza la información obtenida en la exploración para diseñar nuevas y mejores pruebas.

En las PE siempre se debe tomar nota de lo que se hizo y lo que sucedió. Los resultados de las mismas no son necesariamente diferentes de aquellos obtenidos de la prueba con diseño previo y ambos enfoques para las pruebas son compatibles, (Bach 2002).

Por otra parte *Harry Robinson* describe a los probadores como los guías de la prueba, construyendo modelos (los mapas) que son usados posteriormente por otros probadores. “La prueba exploratoria es sumamente útil cuando se tiene un software que nunca se ha probado, es desconocido o inestable. Pero una vez que el producto es más estable, se

necesita tener una forma para aliviar el trabajo intensivo, dígase automatizar el modo de probar", (Robinson *"Exploratory Modeling"*).

En general, la prueba solamente exploratoria puede funcionar para probadores con mucha experiencia. Como ventaja se encuentra que es barato y rápido, como contra que no se tienen medidas de cubrimiento y no deja documentación, (Black 2002). La técnica " Session Based Test Management " de James Bach, (Bach 2002, noviembre), es una forma de hacer prueba exploratoria que lleva muy poca documentación. Su principal ventaja es que es de bajo costo y medible. Su desventaja es que requiere probadores preparados.

De acuerdo a las características del LIPS se ha tenido que adaptar las diferentes ideas y conceptos de PE existentes en la literatura, para dar lugar a un nuevo concepto. Esto no excluye la utilización de las PE tal y como las define James Bach. El propio proceso de pruebas del laboratorio requiere que se utilicen ambas técnicas de pruebas: PE y Pruebas Exploratorias Iniciales (PEI). Para mejor entendimiento y explicación del momento en que se utilizarán las PE en esta investigación, las mismas se dividen en: PEI y Pruebas Exploratorias Paralelas (PEP).

"Las Pruebas Exploratorias Iniciales se definen como el proceso inicial de exploración del producto, diseño y ejecución de pruebas que valida la calidad de este, permitiendo o no el paso a nuevas etapas (iteraciones)".

Estas pruebas como su nombre lo indica serán las primeras que se realicen en el LIPS, antes de la primera iteración de las pruebas funcionales. A su vez las PEP serán aplicadas paralelas fundamentalmente a la segunda iteración de las pruebas funcionales o a cualquier otra iteración en que el especialista de prueba las solicite, éstas se tratan tal y como se definen en la literatura. En la figura que se muestra a continuación se evidencia la aplicación de las PEI y PEP dentro del proceso de pruebas de liberación del DPSW.

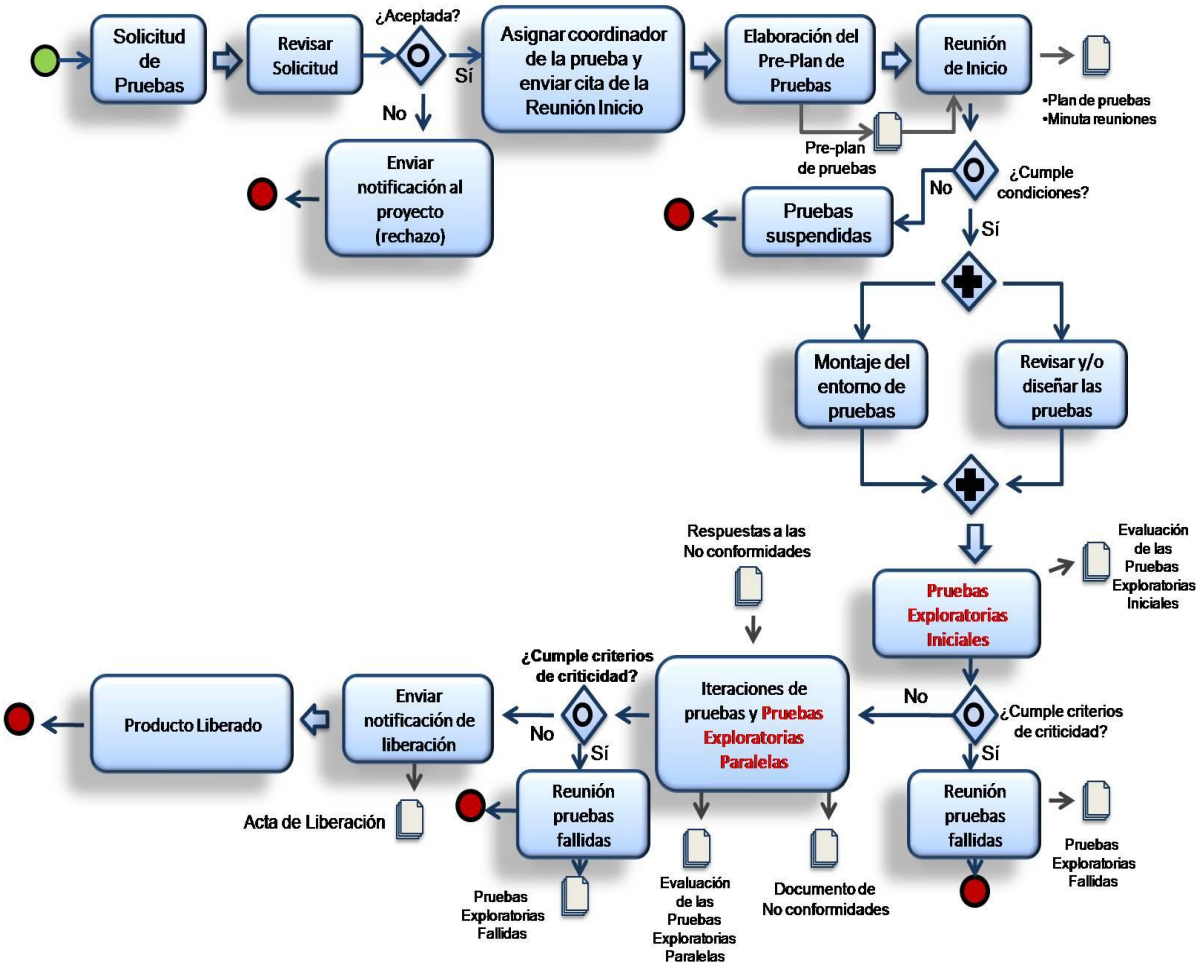


Figura 3: Inserción de las PEI y PEP en el proceso del Laboratorio Industrial de Pruebas de Software.

En esta investigación se definen las PEI y PEP en una estrategia, durante la ejecución de la misma se realizarán diferentes tipos de pruebas en función de las características de los artefactos que se vayan a liberar. De acuerdo a las características del LIPS, la disponibilidad de recursos, el entorno donde se va a implementar la estrategia y el estudio realizado, se seleccionó para dicha estrategia:

1.2.7.1.1.1. Técnicas de muestreo. Selección de la muestra

Hay cuestiones que se deben especificar a la hora de elegir una muestra, (Lagares. P 2001):

- El método de selección de los individuos de la población (tipo de muestreo que se va a utilizar).
- El tamaño de la muestra.
- El grado de fiabilidad de las conclusiones que se van a presentar, es decir, una estimación del error que se cometerá (en términos de probabilidad).

Es de vital importancia una correcta elección de la muestra para que sea representativa para la población, pero ¿cómo se clasifican las diferentes formas de elegir una muestra? Se puede decir que hay tres tipos de muestreo:

1. Muestreo probabilítico: es aquel en el que cada muestra tiene la misma probabilidad de ser elegida.
2. Muestreo intencional u opinático: en el que la persona que selecciona la muestra es quien procura que sea representativa, dependiendo de su intención u opinión, siendo por tanto la representatividad subjetiva.
3. Muestreo sin norma: se toma la muestra sin norma alguna, de cualquier manera, siendo la muestra representativa si la población es homogénea y no se producen sesgos de selección.

En este trabajo siempre se hará muestreo probabilístico, ya que en caso de elegir la técnica adecuada, es el que asegura la representatividad de la muestra y permite el cálculo de la estimación de los errores que se cometen. Dentro del muestreo probabilítico se puede distinguir entre los siguientes tipos de muestreo:

- Muestreo aleatorio con y sin reemplazo.
- Muestreo estratificado.
- Muestreo por conglomerados.
- Muestreo sistemático.
- Otros tipos de muestreo.

Finalmente dentro del muestreo probabilístico, se realizará un muestreo aleatorio sin reposición o sin re-emplazamiento, puesto que una vez elegido un elemento de la muestra no se quiere volver a seleccionar (Lagares. P 2001).

La selección no adecuada de los elementos de la muestra provoca errores posteriores a la hora de estimar las correspondientes medidas en la población, por lo que se hace necesario que la selección se realice con la mayor precisión posible y que realmente sea representativa del artefacto en cuestión. Para ello la autora propone hacer coincidir tres aspectos fundamentales: Principio de Pareto, análisis de los riesgos del producto y aspectos arquitectónicamente más significativos.

Principio de Pareto o la regla del 80:20

Pareto enunció el principio basándose en el denominado conocimiento empírico. Observó que la gente en su sociedad se dividía naturalmente entre los «pocos de mucho» y los «muchos de poco»; se establecían así dos grupos de proporciones 80-20 tales que el grupo

minoritario, formado por un 20% de población, ostentaba el 80% de algo y el grupo mayoritario, formado por un 80% de población, el 20% de ese mismo algo. El principio de Pareto dice que el 20% de cualquier cosa producirá el 80% de los efectos, mientras que el 80% restante sólo cuenta para el 20% de los efectos (Ferris 2010).

No obstante, el principio de Pareto permite utilizar herramientas de gestión, como el diagrama de Pareto, que se usa ampliamente en temas de control de calidad (el 80% de los defectos radican en el 20% de los procesos). Así, de forma relativamente sencilla, aparecen los distintos elementos que participan en un fallo y se pueden identificar los problemas realmente relevantes, que acarrean el mayor porcentaje de errores.

De la misma manera, en el mundo de la ingeniería del software el principio de Pareto puede ser enunciado de diferentes formas (Besterfield 2010):

- Así, por ejemplo, cuando hablamos de los costes de desarrollo podríamos decir que "el 80% del esfuerzo de desarrollo (en tiempo y recursos) produce el 20% del código, mientras que el 80% restante es producido con tan sólo un 20% del esfuerzo".
- Si hablamos de pruebas de software, el principio nos dice que "el 80% de los fallos de un software es generado por un 20% del código de dicho software, mientras que el otro 80% genera tan solo un 20% de los fallos".

Como herramienta de gestión, tiene un peso fundamental en el ámbito de la calidad, siendo uno de los instrumentos básicos que facilita ver de una forma sencilla los distintos elementos que participan en un fallo de una manera cuantitativa, permitiendo centrar el trabajo en aquellos problemas realmente relevantes y que acarrean el mayor porcentaje de errores.

1.2.7.1.1.2. Medir la prueba. Conjunto de métricas asociadas a las PE

La medición permite crear una memoria organizacional y ayuda a contestar distintas preguntas asociadas con cualquier proceso del software. Mejora la planificación del proyecto, permitiendo determinar las fortalezas y debilidades de los procesos y productos. La medición también ayuda, durante el transcurso de un proyecto, a determinar su progreso, tomar acciones correctivas y evaluar el impacto de tal acción. La medición, para ser efectiva debe: enfocarse en objetivos, aplicarse a todos los productos, procesos y recursos del ciclo de vida, interpretarse basándose en la caracterización y conocimiento de la organización, (Basili V. 1994).

Las mediciones permiten cuantificar tanto el proceso como el producto. Proporcionan la visión del desempeño del proceso permitiendo: desarrollar perfiles de los datos de los proyectos anteriores que se pueden utilizar para la planificación y mejora del proceso;

analizar un proceso para determinar como mejorarlo; determinar la eficacia de modificaciones en el proceso, (Pomeroy-Huff M. 2005).

En el proceso de prueba las mediciones pueden ser usadas para, (Kaner C. 2001):

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

En la presente investigación se tendrán en cuenta las métricas de calidad porque son utilizadas una vez que termine de desarrollarse el producto de software, las mismas serán calculadas en las pruebas que se le realicen a los artefactos en el LIPS una vez terminado su desarrollo por los equipos de proyectos de la UCI. Además sirven para evaluar el producto final ante de entregárselo al cliente, teniendo en cuenta los requisitos especificados por el mismo.

A su vez se seleccionan las métricas externas de calidad ya que las mismas están destinadas para medir la calidad del producto software a través de la medición del comportamiento del sistema del cual el software forma parte y pueden ser usadas durante las etapas de pruebas del proceso ciclo de vida y durante cualquier otra etapa operacional.

La utilización de un grupo de métricas en esta investigación tiene como objetivos fundamentales: medir la efeciencia (consumo de tiempo, rendimiento y utilización de recursos de sistemas informáticos), la funcionalidad (idoneidad - estabilidad en las especificaciones funcionales) y confiabilidad (madurez – grado de solución ante fallos totales y erradicación de fallos) durante la ejecución de las pruebas en el LIPS y evaluar el producto final antes de ser entregado al cliente, para así lograr mayor satisfacción de los mismos.

Este tema se trata con mayor profundidad en (Góngora 2012) donde se define el “Catálogo automatizado de métricas de calidad para evaluar los productos en las pruebas del LIPS”.

1.2.7.1.1.3. Herramientas de pruebas no automatizadas

Como herramientas de pruebas no automatizadas se utilizarán las listas de chequeo (LCH) y los casos de prueba (CP).

Las LCH proporcionan un apoyo mayor mediante preguntas que los probadores deben de responder mientras leen el artefacto. Esta técnica facilita listas que ayudan al probador a saber qué tipo de errores buscar. Sólo se aplicarán a los artefactos de documentación.

Los CP son un conjunto de entradas con datos de prueba, unas condiciones de ejecución y unos resultados esperados. Su propósito es identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (casos de uso). Los CP sólo se utilizarán en caso que el artefacto sea de aplicación.

1.2.7.1.1.4. Aspectos generales

Como herramienta para el registro y seguimiento de las NC se utilizará el Redmine, (Lang 2006-2011). En el anexo 8, Tablas 17 y 18 se explican las características de esta herramienta.

En caso de que el artefacto necesite otro tipo de prueba que no sea funcional, se utilizará para la ejecución de las mismas las herramientas descritas en el anexo 8: Herramientas automatizadas.

Se aplican a esta estrategia los criterios de criticidad establecidos por el DPSW, para declarar que un artefacto se encuentra en estado crítico de terminación, dígase, abortado o pruebas fallidas. Son aplicables en tres momentos fundamentales:

- Cuando se hace la reunión de inicio (RI) con el equipo de proyecto.
- Durante el proceso de PEI.
- Durante las iteraciones de prueba (IP).

En el anexo 10 se describen los criterios de criticidad definidos por el DPSW.

1.3. Conclusiones parciales del capítulo I

- La IndSW es una industria creciente a nivel mundial, no obstante aún existen problemas relacionados con la calidad del producto final, la definición y ejecución de procesos y la satisfacción del cliente.
- En la literatura consultada, varios autores consideran que cuando se habla de mejorar la calidad del software la referencia obligada es hacia las pruebas. Aunque no mejoran directamente la calidad del sistema, dan una visión clara de las debilidades observadas y de los riesgos asociados, sobre los cuales se puede trabajar para perfeccionar, catalogándose así como un aspecto crucial dentro de la calidad. Este control de la calidad exige una serie de revisiones y pruebas utilizadas a lo largo del ciclo de

desarrollo para asegurar que el producto cumple con los requisitos solicitados por el cliente.

- En la UCI aunque muchos consideran las PSW de vital importancia para aumentar la calidad de sus productos, según la encuesta aplicada en esta investigación es un proceso que no se ejecuta correctamente, trayendo como consecuencia gastos innecesarios de tiempo, esfuerzo y recursos, y por supuesto, la insatisfacción del cliente.
- Las estrategias y procesos de pruebas a nivel internacional son de obligada referencia, pero hay que llegar a la elaboración de estrategias propias.

Por todo ello es que constituye una necesidad para la UCI y en particular para el DPSW definir y aplicar una estrategia para las PE, la cual incluirá un conjunto de actividades, métricas y herramientas automatizadas que permitirán de forma eficiente mejorar el rendimiento del LIPS.

CAPÍTULO II: ESTRATEGIA DE PRUEBAS EXPLORATORIAS

Introducción

En este capítulo se presenta la Estrategia de Pruebas Exploratorias, la cual constituye una guía para ejecutar diferentes tipos de pruebas de productos de software a una porción de los artefactos, a partir de las especificaciones de requisitos, los casos de uso y/o los diseños de casos de prueba. Estas pruebas serán ejecutadas por un laboratorio dedicado fundamentalmente a las pruebas independientes, el LIPS.

Entre sus principales características se encuentra que es independiente del ciclo de vida utilizado para desarrollar el producto, basado en la selección de la muestra mediante el Principio de Pareto, el análisis de los riesgos fundamentales asociados al artefacto en cuestión y un listado de los aspectos arquitectónicamente más significativos, estos últimos datos proporcionados principalmente por el equipo de desarrollo.

A su vez se realizan un grupo de mediciones para reportar el avance de la estrategia durante la ejecución de las PEI y PEP respectivamente y un grupo de métricas externas que permitirán evaluar la calidad del producto de software durante la prueba.

La estrategia consta de cuatro etapas y puede ser usada para realizar pruebas de software a los artefactos desde su comienzo (artefactos de documentación), cuando se tiene una versión estable del producto y en diferentes iteraciones.

2.1 Definición de la estrategia de pruebas exploratorias

En la Figura 3 se muestran los momentos de ejecución de las PEI y PEP dentro del proceso de pruebas de liberación del DPSW.

Las PE tiene cinco elementos externos básicamente: tiempo, probador, producto, misión e informes. Esta aparente simplicidad permite desplegar una amplia gama de posibilidades para la aplicación de las PE. Una prueba exploratoria se lleva a cabo en un tiempo delimitado, sobre un producto específico, por medio de un/unos probadores en particular, tratando de cumplir una misión, el cual puede reportar el estado y resultados de la misma en cualquier momento.

En la Figura 4 se muestra la estructura básica definida por la autora para elaborar la estrategia de pruebas exploratorias.

Figura 4: Aspectos que conforman la estrategia de Pruebas Exploratorias.

2.1.1 Etapas

Las etapas de la estrategia son el resultado de fusionar distintas propuestas recolectadas de diferentes autores, (Pérez Lamancha 2006) y (Bach 2000). Estas propuestas fueron adaptadas para ser utilizadas en esta investigación. En la Figura 5 se muestran las cuatro etapas de esta estrategia: Exploración del producto (EX), Diseño de la prueba (planificación) (DP), Ejecución de la prueba (EJ), Evaluación de la prueba (EV).

Figura 5: Etapas de la estrategia de Pruebas Exploratorias.

2.1.1.1 Exploración del producto

Se estudian las principales funcionalidades del producto, con el objetivo de definir el alcance de las pruebas y un primer cronograma de los ciclos de prueba (documento Plan de Prueba). A partir de estos datos se realiza la propuesta de servicio de prueba. El especialista realiza una revisión preliminar de la documentación del proyecto, si la misma cumple con los criterios establecidos por el DPSW entonces se sigue con la etapa de Diseño de la prueba, en caso contrario se pasa a la etapa de Evaluación, donde se analizan cuáles fueron los problemas en este proyecto, se emite el informe de prueba detenida o abortada y se archiva para su consulta en futuros proyectos o para sí el proyecto realiza posteriormente una solicitud nueva de prueba (ver anexo 10: Criterios de criticidad).

En esta etapa se estudia principalmente la factibilidad de realizar el proyecto de prueba. En la Tabla 2 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una o de apoyo a la actividad.

Tabla 2: Actividades y artefactos de la etapa Exploración del producto.

Actividades	Artefactos
EX1 - Análisis de la solicitud de prueba	Solicitud de prueba
EX2 - Recepción del producto a probar	Producto de software
EX3 - Recepción de la documentación del proyecto a probar	Artefactos de documentación
EX4 - Revisión preliminar de la documentación del proyecto	Artefactos de documentación de apoyo a las pruebas, (documentos de especificación de CU, diseño de casos de prueba, especificación de requisitos y/o lista de funcionalidades)
EX5 - Definición del servicio de prueba	Pre-plan de prueba
EX6 - Definición de la muestra	Pre-plan de prueba
EX7- Análisis preliminar de riesgos del producto	Riesgos del producto
ET - Estimación de las tareas	Estimación de tareas
RE - Reportar el esfuerzo	Plantilla de esfuerzo

En la Figura 6 se muestra el diagrama de las actividades involucradas en la etapa de Exploración del producto.

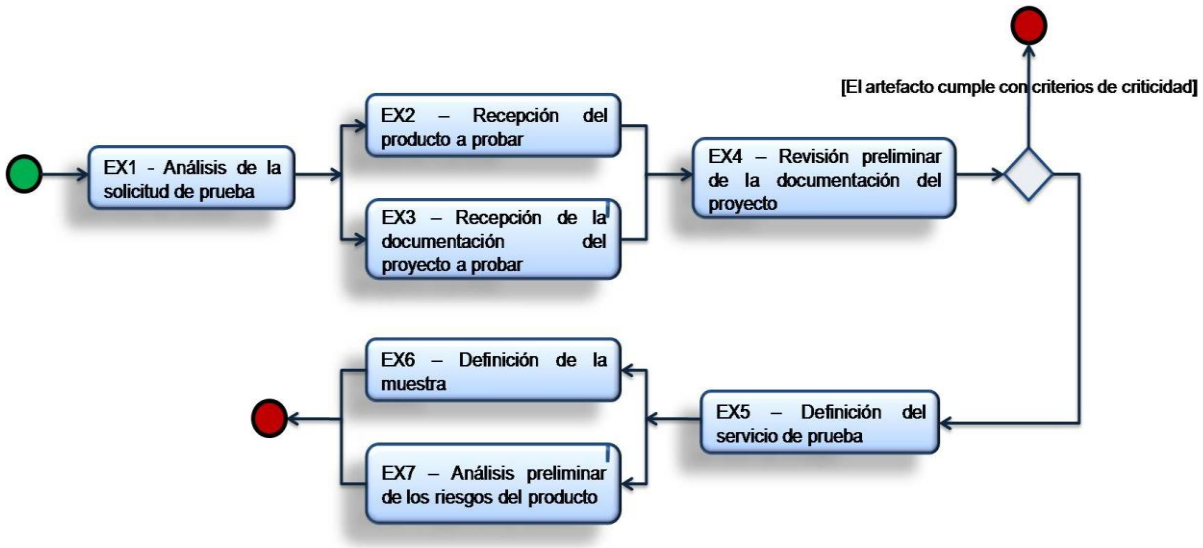


Figura 6: Actividades de la etapa Exploración del producto.

El especialista recibe la asignación del proyecto orientado por el jefe del DPSW. Procede a ejecutar la actividad EX1 - Análisis de la solicitud de prueba, donde se revisa fundamentalmente que estén incluidos todos los aspectos que se piden en la solicitud, (en el anexo 11 se muestra la plantilla de solicitud de prueba). Posteriormente el especialista se pone en contacto con el equipo de desarrollo y le solicita los productos a probar, se ejecutan las actividades EX2 - Recepción del producto a probar y EX3 - Recepción de la documentación del proyecto a probar y se procede a la actividad EX4 - Revisión preliminar

de la documentación del proyecto. El especialista revisa estos artefactos, que estén completos y que cumplan con las reglas para la documentación establecidas por la entidad y/o el proyecto (formato, descripción de las funcionalidades). Estos documentos se revisarán con las listas de chequeo establecidas por el DPSW. En caso que la documentación no cumpla con los criterios determinados por el laboratorio se detiene la prueba y se pasa a la etapa Evaluación de la prueba (EV), donde se emite el informe de pruebas detenidas o abortadas. En caso contrario se continúa con la ejecución de la etapa.

Una vez revisada la documentación del proyecto se procede a la actividad EX5 - Definición del servicio de prueba. En dicha actividad se elabora el Pre-plan de prueba, documento rector de las pruebas. En el mismo se muestra una pequeña descripción del proyecto, se definen los objetivos, el alcance y la estrategia de pruebas de liberación. Se describe el escenario en el que se ejecutarán las pruebas, los roles y responsabilidades, los recursos del sistema y se establece el cronograma de ejecución de las pruebas. Además se incluye en este documento la actividad EX6 - Definición de la muestra, donde se selecciona la muestra para el artefacto en cuestión (esta actividad se realiza según se describe en el 2.1.3, selección de la muestra).

En el caso específico de las PEI y las PEP como su objetivo fundamental es realizarle pruebas a una porción del artefacto, para poder definir el alcance se divide el sistema en componentes, subsistemas o módulos, no todos serán probados con la misma importancia y pueden existir componentes que queden fuera del alcance de las pruebas. Cada componente agrupa varias funcionalidades, se dividen las funcionalidades hasta un nivel en el que sea posible definir el alcance. Para esta selección se tiene en cuenta principalmente la prioridad y complejidad de los requisitos, CU o DCP.

Otros puntos importantes que se deben definir en esta actividad para poder realizar la propuesta de servicio, incluyen:

- Definir los tipos de pruebas que serán tenidos en cuenta para la prueba del producto durante las PEI y PEP, según el modelo de calidad ISO/IEC 9126 (anexo 7, Tabla 16: Tipos de pruebas según la ISO 9126). Se debe definir cuáles de estas características de calidad se debe verificar que estén presentes en el producto y con que relevancia deben ser verificadas.
- Definir el criterio de terminación de las pruebas, este estará dado por el cumplimiento de los criterios de criticidad establecidos en el DPSW, (en el anexo 10 se muestran los mismos).

Para la actividad EX7 - Análisis preliminar de riesgos del producto, es necesario obtener una lista priorizada de las funcionalidades más importantes y riesgosas, así como identificar las funcionalidades que no serán verificadas y el riesgo asociado. A partir del documento de Riesgos del producto, se analizan los riesgos asociados a cada funcionalidad en un trabajo conjunto entre el especialista de calidad asignado a la prueba y el equipo de desarrollo, puesto que éstos últimos conocen las partes del producto que es más probable que presente defectos. La selección de la muestra de las funcionalidades del producto son priorizadas en función del análisis de riesgo realizado {para mayor información de cómo se realizará el análisis de riesgos se puede consultar (Menéndez 2012)}.

2.1.1.2 Diseño de la prueba

El objetivo de esta etapa es diseñar (planificar) el proyecto de prueba, una vez que fue aprobado por el equipo de proyecto en la reunión de inicio. Se define la estrategia de prueba y las funcionalidades a probar en cada ciclo en función del análisis de riesgo del producto y la selección de la muestra. Se genera el Plan de Pruebas que resume toda la información del proyecto de prueba y las decisiones tomadas durante la etapa de diseño de la prueba. De las actividades DP1 - Negociación con el equipo de proyecto, a la DP7 – Planificación de la prueba, se ejecutan durante la reunión de inicio.

En la Tabla 3 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una o de apoyo a la actividad.

Tabla 3: Actividades y artefactos de la etapa Diseño de la prueba.

Actividades	Artefactos
DP 1 – Negociación con el equipo de proyecto. Reunión de inicio	Minuta de reunión
DP 2 – Revisión de la documentación del proyecto	Artefactos de documentación de apoyo a las pruebas, (documentos de especificación de CU, diseño de casos de prueba, especificación de requisitos y/o lista de funcionalidades)
DP 3 – Análisis de riesgos del producto	Riesgos del producto
DP 4 – Refinamiento de la muestra seleccionada	Plan de prueba
DP 5 – Definición de la estrategia de prueba	Plan de prueba
DP 5.1 – Definición de las métricas a calcular en el proceso de prueba	Catálogo de métricas
DP 6 – Definición del tratamiento de las no conformidades	Herramienta Redmine
DP 7 – Planificación de la prueba	Plan de prueba. Cronograma de ejecución de la prueba
DP 8 – Creación del entorno de prueba	Plan de prueba
DP 9 – Creación del proyecto en el repositorio de prueba	Repositorio de prueba
ET - Estimación de las tareas	Estimación de tareas
RE - Reportar el esfuerzo	Plantilla de esfuerzo

En la Figura 7 se muestra el diagrama de las actividades involucradas en la etapa de Diseño de la prueba.

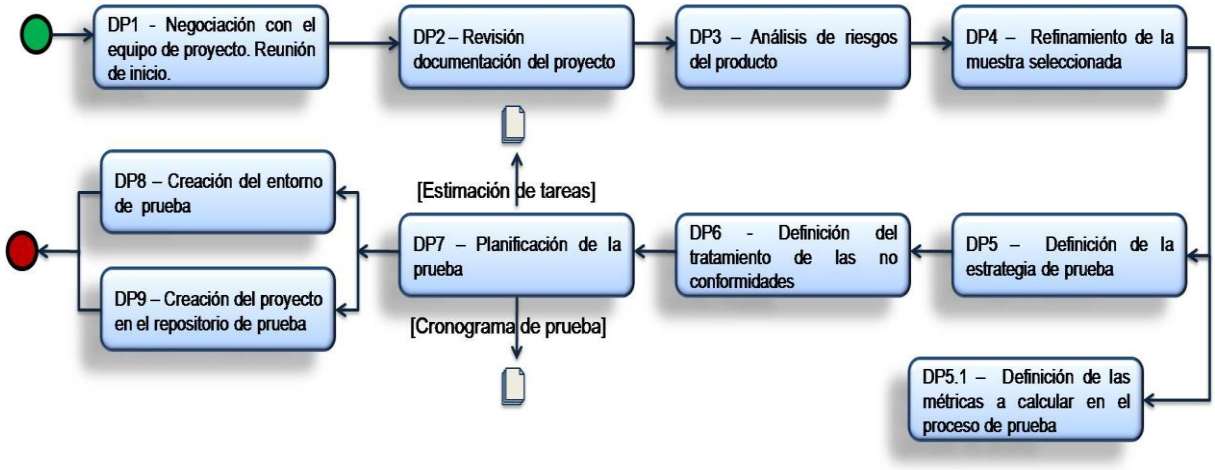


Figura 7: Actividades de la etapa Diseño de la prueba.

La actividad DP1 - Negociación con el equipo de proyecto, tiene como objetivo negociar todos los aspectos que tendrá la prueba desde su comienzo hasta la liberación del artefacto y definir junto con el equipo de desarrollo las funcionalidades a probar y la prioridad con que será probada cada una, para llevar a cabo esta actividad es convocada por el especialista de prueba una Reunión de inicio (estos acuerdos quedan plasmados en el documento: Minuta de reunión). Para ello, en la actividad EX4 - Revisión preliminar de la documentación del proyecto, desde la etapa anterior se comenzó a estudiar cada una de las funcionalidades con que contaba el producto y se revisan en la DP2 – Revisión documentación del proyecto, a profundidad, aquí además de las funcionalidades se analizan todos los documentos de apoyo a las pruebas facilitados por el equipo de desarrollo.

En la actividad DP3 – Análisis de riesgos del producto, se toman los grupos de funcionalidades definidas en la etapa anterior y se realiza el análisis de riesgo de cada funcionalidad del producto, descomponiendo las funcionalidades a nivel macro definida en la etapa Exploración del producto. Puede ocurrir que un grupo de funcionalidades a las que se le asignó prioridad alta durante la etapa anterior, al ser dividido en varias funcionalidades, algunas de ellas sean de prioridad alta, otras de prioridad media y otras queden fuera del alcance de las PEI o PEP.

Durante la actividad DP4 – Refinamiento de la muestra seleccionada, se le presenta al equipo del proyecto en detalles la selección de la muestra que se obtuvo de la etapa anterior y los aspectos que se tuvieron en cuenta para determinar la muestra de acuerdo al tamaño

de su producto. Además durante esta actividad se le pide al proyecto que expongan cuáles son según sus criterios las funcionalidades principales de su producto y los aspectos arquitectónicamente más significativos. Las funcionalidades que ellos determinen prioritarias se le incorporarán a la muestra seleccionada por Pareto, (ver acápite 2.1.3).

En la actividad DP5 - Definición de la estrategia a seguir, se le explica al equipo de proyecto los diferentes ciclos de pruebas a los que serán sometidos sus artefactos, comenzando siempre por las PEI. Se le explica las diferentes etapas con las que cuenta la estrategia de PE, puesto que este tipo de pruebas es una técnica nueva para los desarrolladores. Es de vital importancia que cada especialista de prueba tenga total dominio y conocimiento de esta técnica para que puedan realizar una buena explicación en la reunión de inicio, de ello depende que el proyecto esté de acuerdo con la ejecución de las PEI y PEP.

Aunque las pruebas exploratorias se enfocan fundamentalmente en revisar las funcionalidades del producto, el equipo de proyecto puede solicitar pruebas más técnicas, en este caso se valoraría su posible ejecución. En caso de ser aprobadas estas pruebas se incorporarían al ciclo de ejecución de las pruebas exploratorias y se ajustaría el cronograma de ejecución de las mismas.

Como parte de la Definición de la estrategia a seguir se incluye la actividad DP5.1 – Definición de las métricas a calcular en el proceso de prueba. Al quedar definida el ciclo de prueba exploratoria, con los tipos de pruebas que se le realizarán al software se decide las métricas de calidad que se tendrán en cuenta en las pruebas para la evaluación final del producto de software y del proceso de prueba. En el acápite 2.1.5 se muestra el catálogo de métricas seleccionadas a aplicar durante las pruebas exploratorias.

Mediante la actividad DP6 – Definición del tratamiento de las no conformidades, se le explica al equipo de desarrollo el tratamiento y seguimiento que tendrán las NC durante las PEI y PEP. Esta actividad incluye:

- Explicar la herramienta de seguimiento de no conformidades a usar en el proyecto de prueba, en el DPSW se utiliza el Redmine, (Lang 2006-2011).
- Definir quiénes de las personas del equipo de desarrollo van a ser los encargados de recibir las no conformidades detectadas.
- Mostrar los estados por los que pasará la no conformidad y explicarle cada uno de ellos al equipo de proyecto (los estados por los que pasa una NC definidos por el DPSW se muestran en la actividad EJ4 - Redacción y descripción de las no conformidades).

- Explicar la utilización del Diagrama de Pareto como herramienta de gestión durante las PEI y PEP, que le facilitará al proyecto de forma relativamente sencilla, identificar los problemas realmente relevantes que acarrean el mayor porcentaje de errores, (ver anexo 12).

Finalmente durante la actividad DP7 – Planificación de la prueba se le presenta al proyecto todos los aspectos restantes del plan de prueba y el cronograma de ejecución de la prueba. Este cronograma se define en la reunión de inicio en mutuo acuerdo con el equipo de proyecto, pero durante las pruebas puede presentar cambios. Dicho plan resume toda la información del proyecto de prueba y las decisiones tomadas durante la reunión de inicio. El detalle de la planificación se hace a partir de las estimaciones realizadas en la actividad ET-Estimación de Tareas.

Una vez culminada la reunión de inicio se procede a ejecutar las actividades DP8 – Creación y montaje del entorno de prueba y DP9 – Creación del proyecto en el repositorio de prueba. Estas actividades se pueden realizar en paralelo puesto que las ejecutan diferentes especialistas. La creación del entorno de prueba se realiza según la definida en el plan de prueba y para su montaje se ha decidido virtualizar los servidores, de manera que se logre un uso más eficiente de los recursos con los que se cuenta, procedimiento definido por los especialistas del DPSW, (las herramientas que se utilizan para la virtualización se pueden consultar en (Capote 2011)). Por otra parte en la ejecución de la actividad DP9 se usa el Subversion (SVN) para almacenar y gestionar los proyectos, herramienta muy utilizada para la gestión de configuración, (en (Capote 2011) se muestra la estructura de las carpetas del SVN para los expedientes de prueba).

2.1.1.3. Ejecución de la prueba

El objetivo de esta etapa es generar y ejecutar las pruebas para el artefacto en cuestión. El proyecto de prueba es guiado por el procedimiento establecido por el DPSW (Figura 3 funcionamiento del LIPS) para ejecutar las pruebas de liberación en el LIPS ajustándolo al ciclo de pruebas exploratorias, (la conceptualización e implantación del Laboratorio Industrial de Pruebas de Software se puede consultar en (Capote 2011)).

En la Tabla 4 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una o de apoyo a la actividad.

Tabla 4: Actividades y artefactos de la etapa Ejecución de la prueba.

Actividades	Artefactos
EJ1 - Utilización de la muestra seleccionada	Muestra por Pareto, riesgos del producto y aspectos arquitectónicos más significativos.
EJ2 - Utilización de documentos de apoyo a la prueba	Diseños de casos de prueba, casos de uso, especificación de requisitos, lista de

	funcionalidades. Se utilizan los documentos de apoyo que haya proporcionado el proyecto de acuerdo al artefacto a probar.
EJ3 - Ejecución de la prueba	Historial de pruebas exploratorias No conformidades, Redmine Listas de chequeo Diseño de casos de prueba
EJ3.1 – Recolección de datos para el cálculo de las métricas	Catálogo de métricas
EJ4 - Redacción y descripción de las no conformidades	Estado por los que transitan las no conformidades Clasificación de las no conformidades
EJ5 - Revisión y asignación de las no conformidades	Estados de las no conformidades
EJ6 - Reporte de las no conformidades	Cantidad de no conformidades por tipo de error
ET - Estimación de las tareas	Estimación de tareas
RE - Reportar el esfuerzo	Plantilla de esfuerzo

En la Figura 8 se muestra el diagrama de las actividades involucradas en esta etapa:

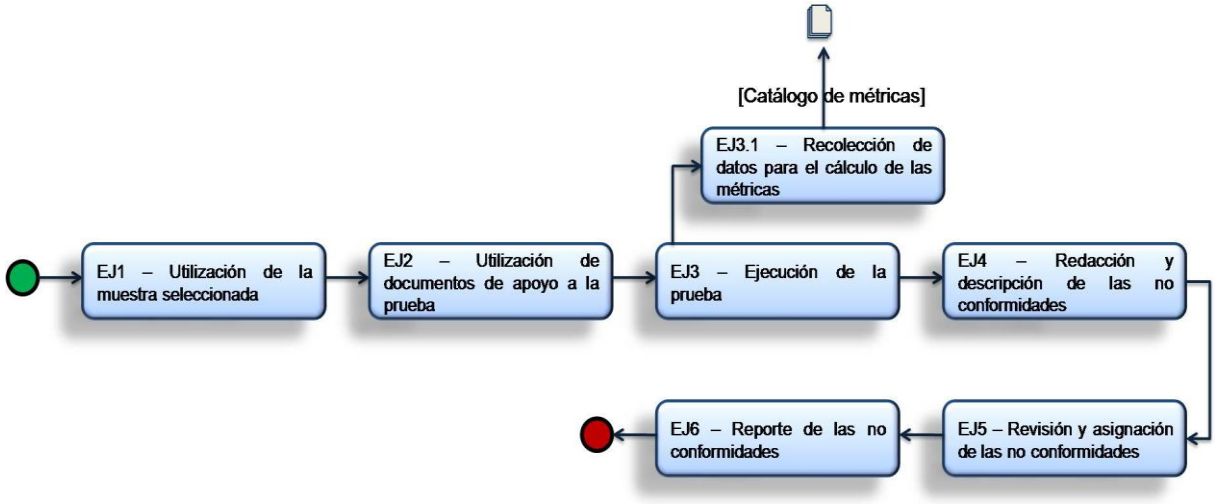


Figura 8: Actividades de la etapa Ejecución de la prueba.

La primera actividad que se ejecuta en esta etapa es la EJ1 - Utilización de la muestra seleccionada, la cual se decidió en las etapas anteriores, donde se hicieron coincidir la selección de la muestra según Pareto (acápite 2.1.3), los riesgos asociados al producto y los aspectos arquitectónicos más significativos, una vez que se tiene este dato es que se toman los documentos de la actividad EJ2 - Utilización de documentos de apoyo a la prueba, para realizarle la prueba al artefacto, comenzando por los diseños de casos de prueba y/o requisitos de complejidad alta y así sucesivamente.

En el caso que los artefactos sean puramente de documentación la muestra se seleccionará solamente por Pareto, tal y como se describe en el acápite 2.1.3 - Selección de la muestra.

Luego se procede a la actividad EJ3 - Ejecución de la prueba, durante esta actividad se va llenando el historial de pruebas exploratorias, documento donde se registra toda la

secuencia de la prueba, en el anexo 13 se muestran los aspectos que componen dicho informe. La prueba la realiza fundamentalmente el especialista asignado al proyecto, puesto que esta técnica de prueba requiere de un amplio conocimiento y se ejecuta en menos tiempo que una iteración de prueba.

Durante la ejecución de las pruebas exploratorias los artefactos se probarán contra las listas de chequeo si son artefactos de documentación y contra los diseños de casos de prueba si son de aplicación. Se utilizarían herramientas automatizadas en caso que en la reunión de inicio se haya acordado incorporar durante el ciclo de pruebas exploratorias la ejecución de otros tipos de pruebas de mayor complejidad (pruebas de seguridad, carga y estrés, volumen, etc.).

Tal y como se menciona en el acápite 1.2.7.1.1 esta estrategia se ejecutará en dos momentos fundamentalmente, al inicio del proceso de pruebas de liberación y/o durante la segunda iteración de las pruebas, esta última a petición del especialista que esté frente al proceso de prueba (ver Figura 3, proceso del LIPS).

La ejecución de las PEP en una segunda iteración es a criterio del especialista de prueba, el mismo puede solicitar este tipo de prueba porque:

- Cree necesario verificar las NC emitidas en iteraciones anteriores, puesto que puede que no hayan sido resueltas en su totalidad.
- El tiempo planificado para la ejecución de la iteración es menor que el que fue planificado con anterioridad.
- Durante la ejecución de la iteración se ha encontrado un grupo importante de NC y cree prudente aplicar las PE a una pequeña porción del artefacto, para de cumplirse algún criterio de criticidad abortar la prueba.
- Se quiere comprobar que la versión del producto que se está probando fue la misma que se revisó en iteraciones anteriores.

Mientras se va ejecutando la prueba se realiza en paralelo la actividad EJ3.1 – Recolección de datos para el cálculo de las métricas, para una vez culminado el ciclo de prueba se emita una evaluación del producto y el proceso de prueba.

En la actividad EJ4 - Redacción y descripción de las no conformidades, se realiza la gestión de las no conformidades que se obtienen durante esta etapa, utilizándose la herramienta Redmine, donde se plasman cada una de las incidencias encontradas, (características de la herramienta en el anexo 8 Tablas 17 y 18 (Lang 2006-2011)). Una vez descrita la(s) no conformidades se selecciona el tipo de error que le(s) corresponde, este se determina según

si el artefacto es de aplicación (A) o documentación (D) y se clasifican en ortografía (A y D), redacción (A y D), error de idioma (A y D), error de interfaz (A), formato (A y D), error técnico (D), excepción (A), funcionalidad (A), validación (A), opciones que no funcionan (A), diseño de caso de prueba (D), correspondencia con otro artefacto (A y D), seguridad (A) y recomendación (A y D).

Las no conformidades a su vez transitan por varios estados (nueva, rechazada, pospuesta, asignada, en curso, detenida, abortada, liberada y cerrada) y son modificados por los roles que se le asignan en el Redmine a los involucrados en la prueba a partir de los permisos que tenga cada cual, ellos son: probador, especialista al frente de la prueba y responsable del proyecto.

Posteriormente se ejecuta la actividad EJ5 - Revisión y asignación de las no conformidades en el Redmine. Aquí se revisa la redacción de las incidencias y se les asignan y notifican al equipo de desarrollo para su corrección.

Finalmente se realiza la actividad EJ6 - Reporte de las no conformidades, donde se obtiene un informe con todos los aspectos importantes de las no conformidades, que permite al equipo de prueba administrar y analizar los incidentes reportados y la tendencia de los mismos en los ciclos de prueba, (comparación con artefactos similares de otros proyectos).

2.1.1.4. Evaluación del producto y de la prueba

Es la última etapa del ciclo de pruebas exploratorias, presenta dos objetivos fundamentales: emitir una evaluación de la calidad del artefacto probado, plasmado en el Informe de evaluación de la prueba definiéndose si pasa o no a la siguiente iteración de prueba y evaluar el proceso de prueba para su mejora almacenando los elementos del proyecto de prueba para su uso en proyectos posteriores.

En la Tabla 5 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una o de apoyo a la actividad.

Tabla 5: Actividades y artefactos de la etapa Evaluación del producto y de la prueba.

Actividades	Artefactos
EV1 - Evaluación del artefacto	Informe de pruebas detenidas Informe de pruebas abortadas
EV 2 - Reporte final del artefacto	Informe reporte final artefacto. Diagrama de Pareto
EV 3 - Archivo de los resultados de la prueba	Repositorio de prueba
EV 4 - Evaluación de la satisfacción del equipo de proyecto	Informe de evaluación de la prueba
EV 5 – Ajustes y mejoras del ciclo de pruebas exploratorias	Resultados del Informe de evaluación de la prueba
EV 6 - Cálculo de métricas	Catálogo de métricas para medir la calidad del proceso de pruebas exploratorias y el producto
EV 7 - Reporte final del ciclo de pruebas	Informe final del ciclo de pruebas exploratorias

exploratorias	
ET - Estimación de las tareas	Estimación de tareas
RE - Reportar el esfuerzo	Plantilla de esfuerzo

En la Figura 9 se muestra el diagrama de las actividades involucradas en esta etapa:

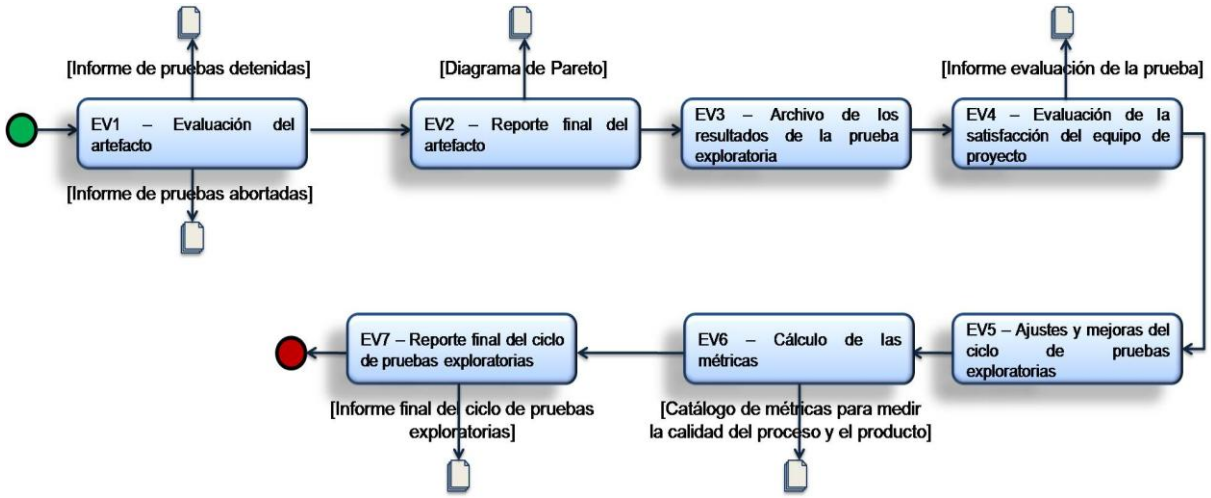


Figura 9: Actividades de la etapa Evaluación del producto y de la prueba.

Una vez culminado el ciclo de pruebas exploratorias comienza esta etapa. La primera actividad que se realiza es la EV1- Evaluación del artefacto, para ello se han denominado por el laboratorio tres categorías, pruebas satisfactorias, abortadas o detenidas. El criterio para determinar estas categorías lo proporcionan el cumplimiento o no de los criterios de criticidad definidos por el DPSW y aprobados por la dirección de la universidad. Los criterios de criticidad establecen los parámetros para declarar un producto del desarrollo de software en estado crítico de terminación, (ver anexo 10).

Si la evaluación de la prueba fue satisfactoria sin errores se pasa directamente a la ejecución de la primera iteración. En caso que la evaluación haya sido satisfactoria pero cuenta con un grupo de no conformidades importantes, los incidentes reportados serán corregidos por los desarrolladores y una vez que regresen al laboratorio se validan las correcciones realizadas de las no conformidades mediante la ejecución de pruebas de regresión para asegurar que los cambios no introducen un comportamiento no deseado o nuevos errores, esto implica seleccionar los casos de prueba, requisitos o funcionalidades a re-ejecutar y ejecutarlos. En caso de encontrar nuevas no conformidades se reportan en la actividad EJ4 - Redacción y descripción de las no conformidades de la etapa anterior. Antes de ejecutar las pruebas de regresión se debe verificar el estado de cada una de las no conformidades en el Redmine.

Como se explicó anteriormente las pruebas una vez culminada su evaluación puede ser: satisfactoria, detenidas o abortadas. A continuación se describen los últimos dos términos:

Prueba Detenida (PD): Se detiene la prueba y se reinicia en la misma actividad que se detuvo. Como máximo una prueba puede estar detenida una semana, este plazo se decide según los motivos que hicieron que se detuviera y debe cumplirse por parte del equipo de desarrollo. En caso de que no se cumpla el tiempo, se declara Prueba Abortada.

Prueba Abortada (PA): Se detiene la prueba y para reiniciarla debe hacerse una nueva solicitud, comenzando el proceso desde el inicio. Como mínimo se necesitan tres días para recomenzar la prueba.

Si la evaluación de la prueba fue abortada, cuando el equipo de desarrollo haya resuelto todas las NC y emitido una nueva versión del artefacto, entonces es que el jefe del proyecto realiza una nueva solicitud de prueba y comienza nuevamente el ciclo de pruebas de liberación iniciando con las PEI, siempre verificando antes la corrección de las no conformidades a través de las pruebas de regresión. En este caso la muestra a seleccionar serían funcionalidades diferentes a las que fueron seleccionadas anteriormente, con la misma concepción de la selección de la muestra descrita en el acápite 2.1.3.

Como la evaluación de la prueba fue abortada se emite el Informe de Pruebas Abortadas, documento diseñado por el DPSW donde se le explica al equipo de proyecto cual fue la muestra que se seleccionó para ejecutar las pruebas y las razones por las cuáles su artefacto no pasó a la primera iteración de las pruebas de liberación (los criterios de criticidad que cumple). Igualmente sucede cuando la prueba es detenida, lo que en este caso se emite el Informe de Pruebas Detenidas.

Culminada la evaluación del artefacto se procede a realizar la actividad EV2 - Reporte final del artefacto, aquí se realiza el informe final de los resultados del tránsito del artefacto por el ciclo de pruebas exploratorias, que contiene la evaluación del artefacto, la sumatoria de la clasificación de las no conformidades por tipo de error y el Diagrama de Pareto. Este último expresado en dos vertientes diferentes:

- Diagrama de Pareto para identificar la incidencia de las no conformidades en el artefacto que se está probando. Identificar los problemas realmente relevantes que acarrean el mayor porcentaje de errores, lo cual le permitirá al equipo de desarrollo saber en qué tipo de error debe trabajar más para solucionar sus mayores problemas, en el anexo 12 se muestra un ejemplo de cómo se utilizará esta herramienta, los pasos para su construcción y los resultados que arroja.

- Diagrama de Pareto para identificar la incidencia de los errores en artefactos del mismo tipo. Dirigir las pruebas hacia las funcionalidades que son más propensas a tener ese tipo de error. Esto permite enfocar la muestra hacia las funcionalidades más vulnerables del producto, logrando mayor eficiencia en la ejecución de la prueba. Para ello se tienen que agrupar los artefactos por similitud (aplicación web, portal web, aplicación desktop, multimedia, artefactos de documentación, etc.) con las no conformidades que se le han encontrado durante las pruebas, esta información se puede obtener del repositorio de prueba del DPSW, e ir enriqueciendo con las pruebas que se vayan ejecutando en el laboratorio, (en el anexo 12 se ejemplifican estos diagramas).

En la ejecución de la actividad EV3 - Archivo de los resultados de la prueba, es utilizado por el LIPS el Subversion (SVN) como se explicó en la actividad DP9 - Creación del proyecto en el repositorio de prueba. Aquí se almacenará toda la documentación que se utilizó para la prueba, la versión de la aplicación que fue probada y el historial de pruebas exploratorias, con el objetivo de mantener un histórico de la prueba para ser consultado en próximos proyectos o iteraciones de prueba y tener un respaldo en caso que el proyecto solicite una revisión.

La actividad EV4 - Evaluación de la satisfacción del equipo de proyecto, tiene como objetivo conocer el grado de satisfacción del equipo de desarrollo con la prueba realizada, para ello se le hace llegar un Informe de evaluación de la prueba, y a partir de sus criterios, en la actividad EV5 - Ajustes y mejoras del ciclo de pruebas exploratorias el equipo de prueba puede mejorar la estrategia de prueba para próximos proyectos.

Una vez culminada la ejecución de la prueba exploratoria se puede realizar la actividad EV6 – Cálculo de las métricas. Se calculan las métricas con los datos recolectados del ciclo de prueba y de las pruebas de regresión (en caso que el artefacto estuviese en segunda iteración o se le comenzó a aplicar nuevamente las PEI), según el tipo de prueba realizado al producto de software y se puede ir observando el porciento de cumplimiento de las características de calidad en el ciclo de pruebas y comprobar si aumentan o disminuyen las mismas. El cálculo de las mismas se realiza mediante la herramienta para la gestión de las métricas, la cual se puede consultar en (Góngora 2012). En el acápite 2.1.5 se definen las métricas que serán utilizadas en esta investigación.

Una vez finalizado el proyecto de prueba, se realiza un reporte que resume el proyecto en su conjunto, actividad EV7 - Reporte final del ciclo de pruebas exploratorias. Este reporte debe indicar el esfuerzo total, el esfuerzo por etapa y por actividad, las desviaciones que ocurrieron respecto a lo planificado y las razones de dichas desviaciones, (desviaciones

respecto a lo planificado en el Plan de Pruebas, principalmente incumplimiento del cronograma de prueba). Las mediciones realizadas durante este ciclo de prueba son sumamente importantes para las estimaciones de los proyectos posteriores.

2.1.2. Aspectos y actividades comunes en las etapas

En cualquier etapa pueden ser aplicados los criterios de criticidad definidos por el DPSW (ver anexo 10) en caso que se cumpla alguno de los criterios se pasa a la etapa Evaluación de la prueba donde se emite un documento con las especificidades de las pruebas, ya sea por haberse detenido la prueba o abortada.

Las siguientes actividades se realizan en todas las etapas de la estrategia de pruebas exploratorias. Son las actividades que tienen que ver con la estimación de las tareas y el reporte del esfuerzo de cada actividad a realizar y su registro una vez que la actividad se realizó.

A partir de esa información se puede estudiar la desviación entre lo planificado y lo real, una forma de anticipar si es posible realizar las pruebas agendadas para el ciclo o si es necesario hacer una nueva planificación del ciclo. Permite conocer además cuáles son los casos de prueba, requisitos o funcionalidades que ocupan el mayor esfuerzo.

2.1.2.1. Estimación de tarea

El especialista de pruebas debe estimar el tiempo que le insumirá cada actividad asignada en el ciclo. Se debe desarrollar la planificación del trabajo antes de comenzarlo, basándose en las actividades del proceso definido y los datos históricos personales.

La estimación es un proceso de aprendizaje, se mejora con la experiencia y los datos históricos. Las estimaciones están sujetas a errores, es mejor estimar por partes, ya que el error total debe ser menor que la suma de los errores de las partes si las partes son estimadas independientemente. Se usan datos históricos para hacer las estimaciones (Pomeroy-Huff M. 2005).

Esta actividad se realiza en conjunto con la actividad DP7 - Planificación de la prueba, donde las estimaciones de cada miembro del equipo son organizadas en el tiempo.

2.1.2.2. Reporte de esfuerzo

Mantener un registro del esfuerzo que insume realizar cada tarea en el ciclo de prueba. El especialista de pruebas debe registrar el tiempo real (horas/persona) que le insumió realizar cada actividad de la estrategia en el ciclo de prueba. En el anexo 14 se muestra la Plantilla de esfuerzo.

2.1.3. Selección de la muestra

Tal y como se explica en el acápite 1.2.7.1.1.1 de este trabajo para la selección de la muestra se harán coincidir tres aspectos fundamentales: Principio de Pareto, análisis de los riesgos del producto y los aspectos arquitectónicamente más significativos.

Durante el desarrollo de este acápite solamente se expondrá la selección de la muestra de los artefactos que con mayor frecuencia son probados en el LIPS y que están definidos como entregables al cliente en la última versión del documento "Clasificación de Documentos", emitido por Calisoft. En el anexo 15 se profundiza en este tema y se presenta la muestra de otro grupo de artefactos.

2.1.3.1. Principio de Pareto

Una vez aplicado Pareto y por ende obtenido el tamaño de la muestra, se procede a seleccionar los elementos que van a formar la muestra, para ello se utilizará una Lista con la prioridad o Criterios de selección. Dado que cada artefacto tiene sus propias características es que se decide a cuál categoría pertenece, igualmente se le realiza un tratamiento diferente a los artefactos de aplicación y documentación.

2.1.3.1.1. Criterios de selección

Artefactos de aplicación: portal web, aplicación web, aplicación desktop, multimedia, prototipos documentados y aplicaciones en ambientes reales - realidad virtual.

Los artefactos de aplicación que se hayan hecho con la metodología RUP, se le seleccionará la muestra en dependencia de la descripción de los documentos:

1. Especificación de requerimientos.
2. Especificación de CU.

Nota: En el caso de los artefactos de tipo aplicación que no tengan CU, el equipo de desarrollo debe presentar el documento especificación de requisitos, especificando la complejidad y prioridad de cada requisito. Igualmente en el documento de especificación de CU deben presentar la prioridad de los mismos. Se recomiendo utilizar el método evaluación por complejidad de los requisitos, definido por Calisoft.

Estos artefactos se revisan contra los diseños de CP perteneciente a los requisitos seleccionados para la muestra.

Artefactos de documentación: manual de usuario, diseño de caso de prueba, especificación de requerimientos y especificación de CU.

En el artefacto manual de usuario, se revisarán las páginas que describan la muestra seleccionada para la aplicación.

Para la selección de la muestra del artefacto especificación de requerimientos, se tiene en cuenta la complejidad y prioridad de los requisitos, en caso que estos aspectos no se especifiquen en el documento solicitud de pruebas, en la reunión de inicio se le pide al equipo de desarrollo que le asigne las prioridades a los requisitos, luego se calcula por Pareto el tamaño de la muestra, o sea, el 20 % del total (comenzando por los de complejidad alta) y se seleccionarán los requisitos que van a formar parte de la muestra, teniendo en cuenta su prioridad.

Para la selección de la muestra del artefacto especificación de CU, se calcula por Pareto el tamaño de la muestra, o sea, el 20 % del total y se seleccionarán los CU teniendo en cuenta su prioridad.

Para la selección de la muestra del artefacto diseño de casos de prueba, se utilizan los mismos CU que hallan sido seleccionado en la muestra del documento especificación de CU, y si no trabajarán con CU, entonces se utilizaría la muestra seleccionada para el artefacto especificación de requerimientos.

Como se mencionó en el capítulo anterior se utilizará como herramienta no automatizada para revisar los artefactos de documentación diferentes secciones de las listas de chequeo diseñadas por Calisoft. Para revisar los documentos anteriores (especificación de requerimientos, especificación de CU y diseño de casos de prueba), se utilizarán de manera general las secciones de la lista de chequeo que se muestran en el anexo 15.

2.1.3.1.2 Lista de prioridades

Para hacer la lista de prioridades se debe usar Pareto, para ello se definieron todos los acápites del documento y luego se seleccionó una muestra de 20 proyectos y se le realizaron las pruebas funcionales a la primera iteración para ver cuantas NC habían en cada acápite, luego se ordenaron de mayor a menor. Posteriormente se calculó el 20 % del total de acápites y eso dio un número, el cual representar el 80 % del porcentaje real de las NC que se detectaron. En el anexo 16 se muestra el diseño de las tablas para confeccionar la lista de prioridades y la validación de los expertos.

Artefactos de Documentación: modelo de despliegue, glosario de términos.

Características: Se seleccionará una cantidad de acápites al documento según Pareto, para ello se conforma una lista de prioridades, en dependencia de los criterios de los expertos.

Nota: Se revisarán sólo las páginas del documento que se seleccionen según la cantidad que define Pareto. Las páginas serán seleccionadas según una encuesta que se le realizó a varios expertos.

Validación según expertos: Se realizó una encuesta a siete expertos, según la prioridad que ellos le asignaron a los diferentes acápites de los documentos antes mencionados, luego se hizo un promedio de los mismos y se le asignó un orden de prioridad a cada acápite de cada documento. A continuación se muestran las tablas con estos resultados.

Documento modelo de despliegue:

Tabla 6: Resultado ordenado de los acápites del documento modelo despliegue.

Acápite	Promedio
Diagramas de despliegue	1,6
Descripción de nodos	2,6
Propósito	3,8
Alcance	4,2
Definiciones, acrónimos y abreviaturas	4,2
Introducción	4,4
Referencias	5,2

El 20 % de la cantidad total de acápites es 1.4, redondeando por defecto es aproximadamente 1. Por lo que la muestra significativa que se escogerá es el siguiente acápite: Documento diagramas de despliegue. Para revisar estos acápites se utilizarán las secciones de la lista de chequeo que se muestran en el anexo 15.

Documento glosario de términos:

Tabla 7: Resultado ordenado de los acápites del documento glosario de términos.

Acápite	Promedio
Definiciones	3,8
Expediente de proyecto	5
Casos de uso	5,2
Lineamientos mínimos de calidad	6,4
Casos de uso significativos para el negocio	6,6
Prototipo	6,8
Casos de uso arquitectónicamente significativos	6,8
Pruebas de liberación	8,6
Pruebas de aceptación	9
No conformidades	9
Propósito	9,6

Solicitud de cambio	9,6
Introducción	11,2
Alcance	11,4
Definiciones, acrónimos y abreviaturas	11,4
Referencias	12,2

El 20 % de la cantidad total de acápites es 3.2, redondeando por defecto es aproximadamente 3. Por lo que la muestra significativa que se escogerá serán los acápites: definiciones, expediente de proyecto y casos de uso. Para revisar estos acápites se utilizarán las secciones de la lista de chequeo que se muestran en el anexo 15.

2.1.3.2. Pruebas basadas en los riesgos del producto

Como se planteó en el Capítulo I la prueba exhaustiva no es posible, por lo que, para cada producto se debe realizar la prueba más efectiva y menos costosa que asegure que el producto es suficientemente confiable, seguro y cumple con los requerimientos de los usuarios. Dado que el tiempo para realizar la prueba nunca es suficiente, deben definirse prioridades. Por tanto, una de las estrategias que asume la autora es tomar la decisión de qué probar basándose en los riesgos.

Se identifican las partes del sistema que en caso de fallar tienen las consecuencias más serias y aquellas que tienen mayor frecuencia de uso, ya que si una parte del sistema es usada frecuentemente y tiene un error, el uso frecuente hace que se tengan grandes posibilidades de que la falla aparezca.

Dado que el análisis de riesgos va a detectar los problemas que podría presentar el sistema durante su puesta en marcha y posteriormente; este método podría identificar la probabilidad de ocurrencia del fallo y el impacto que podría tener sobre el producto. De esta forma arrojaría un resultado más significativo en cuanto a las no conformidades que podrían resultar de las pruebas exploratorias y funcionales y serviría como un argumento fuerte a la hora de aplicar los criterios de criticidad y tomar las decisiones respectivas a la continuidad del proceso de pruebas o la detención del mismo.

La magnitud de un riesgo es proporcional a la probabilidad y el impacto del problema. Cuanto más probable es que el problema suceda, y más alto el impacto, más alto es el riesgo asociado a ese problema. Existen dos enfoques heurísticos para el análisis de riesgo, uno "desde adentro hacia fuera" y otro "desde afuera hacia adentro". Son acercamientos complementarios, cada uno con sus fortalezas. El enfoque de adentro hacia fuera pregunta: "¿qué riesgos se asocian a esta funcionalidad?", mientras que el enfoque de afuera hacia

adentro es el opuesto "¿qué funcionalidades se asocian a esta clase de riesgo?" (Beck 1999), (Pérez Lamancha 2006).

La autora usa el enfoque “desde adentro hacia fuera” para realizar el estudio de riesgos del producto, el cual se describe a continuación:

Enfoque desde adentro hacia fuera:

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

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

El análisis de riesgo es un método heurístico, por lo que se podrían estar dejando de lado funcionalidades importantes para el negocio. Para evitar que esto ocurra, esta estrategia realiza además de las pruebas basadas en riesgos, el análisis de los aspectos arquitectónicamente más significativos para el software y el Principio de Pareto para obtener mayor representatividad en la muestra, esta técnica puede ser consultada en el 2.1.3.1.

El cómo determinar los riesgos se trata con mayor profundidad en (Menéndez 2012), por lo tanto, es un documento de obligada consulta puesto que las pruebas a partir del análisis de los riesgos del producto es una técnica utilizada por la autora para la selección de la muestra durante las PEI y PEP.

2.1.3.3. Aspectos arquitectónicos más significativos

Este enfoque requiere conocimiento técnico, pero no necesariamente de quien prueba. Se le pide al equipo de desarrollo en la reunión de inicio que identifiquen los aspectos que preocupan respecto al producto, para determinar las funcionalidades a tener en cuenta durante las pruebas y se discuten cómo insertarlos a la muestra seleccionada con anterioridad.

Una sesión como esta hace que quien prueba entienda mejor las funcionalidades del producto, obtenga una lista de riesgos específicos y de estrategias asociadas de la prueba. Este enfoque requiere habilidades de comunicación y voluntad de cooperación.

2.1.4. Equipo de pruebas independiente, los especialistas del DPSW

En esta investigación la motivación de contar con un equipo de prueba independiente del desarrollo surge para mitigar la falta de objetividad al realizar la prueba que puede ocurrir cuando el proyecto que desarrolla es la misma que escribe las pruebas. Esto, sumado a las presiones institucionales por salir al mercado hace que en muchos casos la calidad del software se conozca recién cuando se pone en producción. La realización de pruebas independientes también implica beneficios desde el punto de vista de los objetivos de las organizaciones involucradas. Para la organización que realiza las pruebas (LIPS) el objetivo es el de encontrar defectos, para los equipos que desarrolla el producto (proyectos productivos) el objetivo es construirlo.

Las ventajas que se obtienen al tener una organización de prueba independiente de las de desarrollo son: motivación en el proceso de pruebas, oposición de intereses con la organización de desarrollo, separación del proceso de pruebas del control gerencial de la organización de desarrollo y el conocimiento especializado que la organización independiente tiene respecto a las pruebas, (Myers G. 2004).

Dentro de las desventajas se encuentra que la organización de pruebas independiente puede no tener el conocimiento necesario del dominio del negocio, lo que puede llevar a que olvide aspectos importantes o los subestime. Además, si la curva de aprendizaje del producto es elevada, la transmisión de conocimiento puede ser demasiado costosa y se corre el riesgo de que con el tiempo el equipo de pruebas independiente sea el único que conoce como probar el producto.

La eficacia de encontrar defectos puede ser mejorada usando probadores independientes, las opciones para la independencia son, (Kaner C. 2001):

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

El éxito o fracaso de las PE están íntimamente ligados con lo que sucede en la mente del probador. Algunas habilidades deseables en los probadores son la capacidad de analizar el producto y evaluar los riesgos, utilizar herramientas y tener un pensamiento crítico acerca de lo que se sabe al momento de diseñar las pruebas. También es importante que los probadores tengan la capacidad de distinguir lo observado de lo inferido; ya que las

inferencias podrían inducirlos a no realizar pruebas que evidencien vulnerabilidades del producto.

El uso de heurísticas desempeña un rol importante en la producción de ideas variadas. Una heurística es un mecanismo mental para generar pruebas variadas y acordes a las características del producto que se está probando, por lo que es deseable que los probadores las tengan presentes al momento de realizar las PE.

2.1.5. Métricas de calidad

Existe un grupo de métricas internas que aunque son aplicadas fundamentalmente a los productos intermedios que se desarrollan a lo largo del ciclo de vida de desarrollo, la autora considera que las mismas pueden ser adaptadas para reportar el avance en la ejecución de las pruebas en evaluaciones realizadas al final del proceso o ciclos de pruebas (PEI, PEP, iteraciones), por eso es que se toma la decisión de incluirlas como parte de la estrategia desarrollada en esta investigación.

En la Tabla 33 del anexo 17 se listan las mediciones al proceso que se tienen en cuenta en esta investigación. Dichas mediciones fueron recolectadas de: (ISTQB. 2005), (Kaner C. 2001) y (Kit 1995).

La ISO 9126 parte 2 (NC-ISO-IEC-9126-2 2005) contiene un conjunto de métricas externas que permiten evaluar calidad del producto de software durante la prueba, sin asignar rangos de valores a las métricas que propone ya que son específicas para cada producto, en dependencia de su categoría, nivel de integridad y necesidad del usuario final. Las métricas están distribuidas por las seis características que recoge la parte 1 Modelo de Calidad, (NC-ISO-IEC-9126-1:2005) ellas son: Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad. En el anexo 18 se muestran las caracterísiticas y subcaracterísticas del Modelo de Calidad.

Para esta investigación solamente se tendrán en cuenta las métricas asociadas a las carácterísticas de calidad de **eficiencia** con sus subcaracterísticas, **funcionalidad** midiendo la idoneidad y **confiabilidad** teniendo en cuenta la madurez. Para mayor profundización se puede consultar (Góngora 2012).

2.1.5.1. Característica de eficiencia. Métrica de eficiencia

Una métrica de eficiencia externa debe ser capaz de medir atributos tales como el consumo de tiempo y el comportamiento de utilización de recursos de sistemas informáticos, incluyendo el software durante las pruebas o las operaciones.

- Métricas de comportamiento en el tiempo

Una métrica externa de comportamiento en el tiempo debe ser capaz de medir atributos tales como el comportamiento temporal del sistema informático incluyendo el software durante las pruebas o las operaciones.

La métrica de calidad que conforma al catálogo de métricas de la subcaracterística comportamiento en el tiempo es: Métrica # 26. Tiempo de respuesta.

- Métricas de rendimiento

Una métrica externa de rendimiento debe ser capaz de medir atributos tales como el comportamiento utilizando los recursos del sistema informático incluyendo software durante las pruebas o las operaciones.

La métrica de calidad que conforma al catálogo de métricas de la subcaracterística rendimiento es: Métrica # 27 Rendimiento.

- Métricas de eficacia

Una métrica de eficiencia debe ser capaz de medir el cumplimiento de atributos tales como el número de funciones, o por otros acontecimientos de los problemas de cumplimiento, que puede ser el producto de software que no cumplan con las normas, convenciones o regulaciones relativas a la eficiencia.

La métrica de calidad que conforma al catálogo de métricas de la subcaracterística eficiencia es: Métrica # 28: Cumplimiento de la eficiencia.

2.1.5.2. Característica de funcionalidad. Métrica de funcionalidad

Las métricas externas de funcionalidad deben ser capaces de medir de un atributo como es el comportamiento funcional del sistema en el cual el software está presente. En esta investigación solamente se tiene en cuenta la métrica de idoneidad.

- Métricas de idoneidad

Las métricas externas de idoneidad deben ser capaces de medir de un atributo como es la ocurrencia de un funcionamiento insatisfactorio o la ocurrencia de una operación insatisfactoria.

Un funcionamiento u operación insatisfactoria puede ser:

- Funcionamiento u operación que no se desempeña de la forma especificada en el manual de usuario o la especificación de requisitos.
- Funcionamiento u operación que no provee una salida aceptable o razonable al tomar en consideración un objetivo específico de las tareas del usuario.

Las métricas de calidad que conforman al catálogo de métricas de la sub-característica idoneidad es entre otras: Métrica # 4: Estabilidad en las especificaciones funcionales.

2.1.5.3. Característica de confiabilidad. Métrica de confiabilidad

Las métricas externas de confiabilidad deben ser capaces de medir atributos relacionados con el comportamiento del sistema del cual el software forma parte durante la ejecución de las pruebas para indicar la magnitud de la confiabilidad, o sea, seguridad de funcionamiento del software durante la operación del sistema, con las que en la mayor parte de los casos no se distingue entre el software y el sistema. En esta investigación solamente se tiene en cuenta la métrica de madurez.

- Métricas de madurez

Las métricas externas de madurez deben ser capaces de medir de un atributo como la exención de fallas en el software, causados por la ocurrencia de fallos existentes en el propio software.

Las métricas de calidad que conforman al catálogo de métricas de la sub-característica madurez son entre otras: Métrica # 12: Grado de solución ante fallos totales y Métrica # 14: Erradicación de fallos.

En la Tabla 34 del anexo 19 se listan las mediciones al producto que se tienen en cuenta en esta investigación, (Métricas externas de calidad. Características de eficiencia, funcionalidad y confiabilidad). Además se presentan en las Tablas de la 35 a la 39 las métricas externas definidas y seleccionadas según la característica de calidad a la que pertenezcan, guiado por la ISO 9126-1 parte 1: Modelo de Calidad con el método de aplicación y el tipo de medida de cada una de las variables que intervienen en la fórmula para el cálculo de la misma.

Las métricas que se mencionan a continuación son las que pueden ser calculadas al terminar el ciclo de pruebas exploratorias:

- Métrica # 26. Tiempo de respuesta.
- Métrica # 27 Rendimiento.
- Métrica # 28: Cumplimiento de la eficiencia.

Las métricas que se mencionan a continuación son las que pueden ser calculadas al terminar las pruebas de regresión de una iteración:

- Métrica # 4: Estabilidad en las especificaciones funcionales.
- Métrica # 12: Grado de solución ante fallos totales.
- Métrica # 14: Erradicación de fallos.

2.2. Conclusiones parciales del capítulo II

- Se definió por la autora la Estrategia de Pruebas Exploratorias la cual está formada por cuatro etapas: exploración del producto, diseño de la prueba, ejecución de la prueba y evaluación de la prueba.
- La realización de diagramas de actividades para cada etapa de la estrategia permite explicar mejor la interacción entre las distintas actividades de la misma.
- La definición por la autora de la selección de la muestra haciendo coincidir tres aspectos fundamentales: Principio de Pareto, análisis de los riesgos del producto y los aspectos arquitectónicamente más significativos, permite incidir sobre las funcionalidades principales del artefacto y garantiza que la muestra que se seleccione sea realmente representativa del producto a probar.
- La inclusión del Diagrama de Pareto como herramienta de análisis de datos en la determinación de la causa principal durante un esfuerzo de resolución de problemas. Su representación gráfica ha facilitado su implantación y comprensión principalmente para el equipo de desarrollo, proporcionándoles una técnica para identificar las funcionalidades que son más propensas a fallar, permitiendo enfocar su atención en las no conformidades que son las responsables por la mayor parte del impacto negativo sobre la calidad del artefacto.
- La definición y selección de las métricas de calidad teniendo en cuenta las características de calidad según la ISO 9126-1 Parte 1: Modelo de Calidad, permite tener una referencia de cuál es el por ciento de cumplimiento que tienen los software de las características de calidad, ayudando a los directivos a la toma de decisiones y a mejorar la calidad del proceso de pruebas y del producto final. Además estás métricas se incluyen en esta estrategia con la idea de brindarle al equipo de desarrollo una evaluación completa de su producto.
- La Estrategia de Pruebas Exploratorias en el DPSW constituye una técnica para mejorar el rendimiento del Laboratorio Industrial de Pruebas de Software. Su integración con otros tipos de pruebas y con diferentes estrategias y procedimientos definidos por especialistas del LIPS contribuye a perfeccionar y fortalecer la calidad con que se ejecutan las pruebas en el laboratorio.

CAPÍTULO III: APLICACIÓN PRÁCTICA Y VALIDACIÓN DE LA ESTRATEGIA DE PRUEBAS EXPLORATORIAS

Introducción

En el presente capítulo se realiza un estudio de los resultados alcanzados con la implantación de la estrategia de pruebas exploratorias como parte del proceso de pruebas de liberación del DPSW de Calisoft. El estudio se basa en los resultados obtenidos en las pruebas exploratorias a partir de la variable y sub-variables definidas en la hipótesis y en la introducción de la tesis, realizando comparaciones de los datos antes de implantada la estrategia y posterior a su puesta en marcha.

Se presentan los resultados de aplicar las métricas propuestas en la estrategia a través de verificaciones realizadas durante las pruebas a proyectos seleccionados, logrando medidas de la efectividad del proceso de prueba a partir de la evaluación de los productos liberados, teniendo en cuenta el cumplimiento de las características de calidad de la ISO 9126 Parte 1: Modelo de calidad. Los datos para definir la media fueron tomados de la tesis “Catálogo automatizado de métricas de calidad para evaluar los productos en las pruebas”, la cual se rige por la metodología propuesta por la IEEE Standard for a Software Quality Metrics Methodology.

Finalmente se muestran los resultados obtenidos de la aplicación de los métodos científicos (entrevistas y encuesta) para validar la propuesta presentada en esta investigación.

3.1. Validación de la propuesta

Para validar la propuesta se tendrá en cuenta la variable definida en el problema (eficiencia) y sus sub-variables (tiempo, rendimiento, eficacia). Según la Real Academia de la Lengua Española (RAE), **eficiencia** es: capacidad de disponer de alguien o de algo para conseguir un efecto determinado. A su vez consiste en la medición de los esfuerzos que se requieren para alcanzar los objetivos. El costo, el tiempo, el uso adecuado de factores materiales y humanos y cumplir con la calidad propuesta. **Rendimiento**: producto o utilidad que rinde de algo o de alguien y proporción entre el producto o el resultado obtenido y los medios utilizados. **Eficacia**: mide los resultados alcanzados en función de los objetivos que se han propuesto, presuponiendo que esos objetivos se mantienen alineados con la visión que se ha definido. Se encuentra en el equilibrio entre la producción de los resultados deseados y la capacidad de producción.

Dichas definiciones se complementan, fundamentando que mediante la disposición de un estado actual se logre el estado deseado, con la utilización de un grupo de recursos. Para

demostrar el antes y después en el DPSW, como se explicó en la introducción de esta investigación se desglosan las variables mencionadas anteriormente en: tiempo de ejecución de las pruebas, optimización de recursos, esfuerzo dedicado a la actividad de prueba y calidad de las pruebas.

3.1.1. Calidad de las pruebas

La calidad es un factor intangible, sin embargo en el caso específico de esta investigación se demostrará a través de un análisis comparativo entre la cantidad de no conformidades detectadas durante la ejecución de las pruebas exploratorias iniciales y la primera iteración de prueba que se evidencia un aumento de la calidad en la ejecución de las pruebas en el laboratorio.

3.1.1.1. Comparación entre la cantidad de no conformidades detectadas durante la ejecución de las pruebas exploratorias iniciales y la primera iteración de prueba

Uno de los aspectos importantes que propone la estrategia con su aplicación es la disminución de la cantidad de NC que se detectan en las iteraciones de prueba una vez ejecutadas las PE, para demostrarlo, es necesario establecer una comparación entre las NC de los artefactos que se le han aplicado las PEI y artefactos que han sido probados en el LIPS, estos últimos sin la ejecución de las PE. Los aspectos comparativos se establecen entre los resultados de la primera iteración una vez ejecutadas las PEI y la primera iteración de prueba sin haberse ejecutado las PEI, los datos se obtuvieron haciendo búsquedas en los repositorios del DPSW y entrevistas a profundidad a los especialistas involucrados en la liberación de dichos artefactos. Fueron seleccionados en total 24 proyectos reales que han sido probados en el LIPS, de ellos a 12 se le ejecutaron PEI y 12 no, para la comparación se hizo coincidir los proyectos que tuviesen artefactos iguales y CP, DCP o requisitos similares, tanto en cantidad como en complejidad.

A continuación se presenta la Tabla 8 con los resultados de la búsqueda y la comparación entre la cantidad de NC de los artefactos. En la última columna se muestra la resta de las NC de la primera iteración (sin ejecutarle PEI) y las NC detectadas durante las PEI. En el anexo 20 se presentan los mismos datos de la Tabla 40 pero con la descripción de las NC por tipo de error.

Tabla 8: Comparación entre la cantidad de no conformidades detectadas durante la ejecución de las PEI y la primera iteración de prueba.

En las PEI					Sin ejecutar PEI (1ra it)				Después de las PEI (1ra it)				Resultados
Proyecto	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Cant. total NC**	**Estado**	**Proyecto**	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Cant. Total NC – 1ra it**	**Proyecto**	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Cant. total NC - PEI**	**Diferencia / Cant. total NC- 1ra it y Cant. Total NC- PEI**
A	a1	28 pág.	20	P. abortadas	B	b1	16 pág.	25	A	a1	28 pág.	18	**7 NC**
C	c1	45 CP, (11 CP muestra)	6	P. satisfactorias	D	d1	41 DCP	94	C	c1	45 CP	90	**4 NC**
E	e1	145 CP, (29 CP muestra)	42	P. abortadas	F	f1	156 CP	115	E	e1	145 CP	62	**53 NC**
G	g1	9 CP (2 CP de muestra)	21	P. abortadas	H	h1	14 CP	42	G	g1	9 CP	5	**37 NC**
I	i1	4 CP (muestra los escenarios más complejos)	6	P. satisfactorias	J	j1	6 CP	24	I	i1	4 CP	5	**19 NC**
K	k1	23 CP (6 CP muestra)	14	P. satisfactorias	L	l1	29 CP	48	K	k1	23 CP	32	**16 NC**
M	m1	73 CP (19 CP de muestra)	5	P. satisfactorias	N	n1	65 DCP	171	M	m1	73 CP	20	**151 NC**
O	o1	23 CP (5 DCP muestra)	14	P. satisfactorias	P	p1	21 DCP	106	O	o1	23 CP	32	**74 NC**
Q	q1	4 CP (muestra los escenarios más complejos)	6	P. satisfactorias	R	r1	6 CP	24	Q	q1	4 CP	5	**37 NC**
S	s1	7 CP (2 CP de muestra)	3	P. satisfactorias	T	t1	6 CP	34	S	s1	7 CP	6	**28 NC**

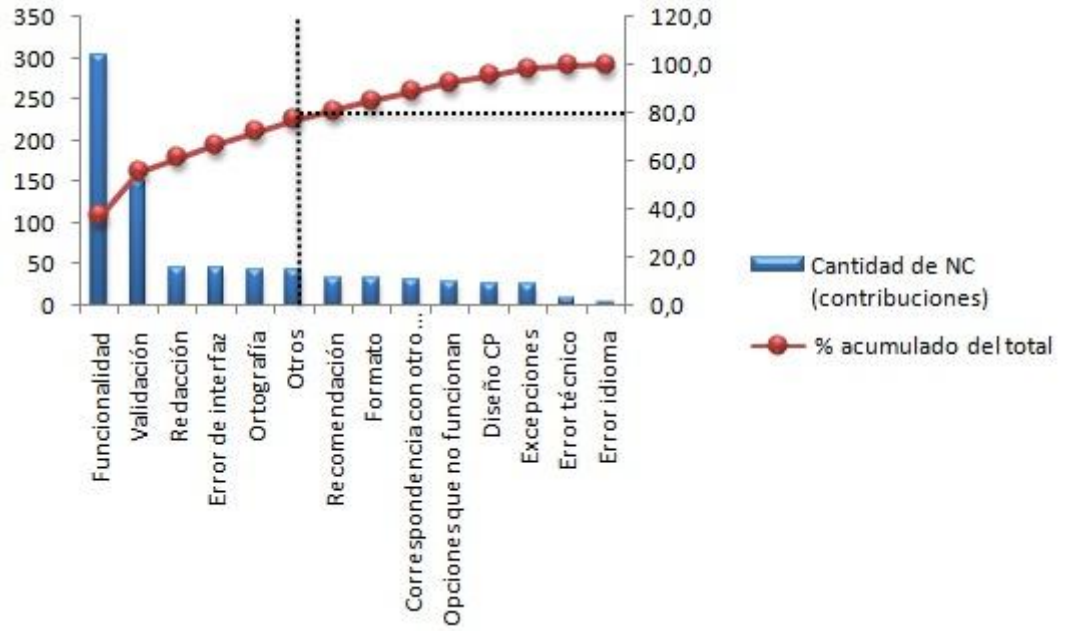
U	u1	63 CP (16 CP de muestra)	26	P. abortadas	V	v1	45 CP	44	U	u1	63 CP	33	**11 NC**
W	w1	45 CP (11 CP de muestra)	6	P. satisfactorias	X	x1	54 CP	97	W	w1	45 CP	90	**7 NC**

Como se puede observar en la tabla anterior, con la aplicación de la estrategia de PE propuesta en esta investigación, a los artefactos se le detectan en la primera iteración mayor cantidad de NC que cuando se le aplican las PEI. Los resultados de la diferencia representan que los artefactos llegan con mejor calidad y por ende, menos NC a la primera iteración de pruebas cuando se le han realizado las PEI, sin embargo, es de vital importancia realizar una buena ejecución de las PE, de ello depende que luego en la 1ra iteración no se encuentren numerosas NC. Por ejemplo, en el proyecto C que se le aplicó la estrategia de PE se encontraron 90 NC en la 1ra iteración y al proyecto D al cual no le aplicaron PE le detectaron 94 NC, aunque en el proyecto D se detectaron más NC que en el C, según criterio de la autora las PE no se ejecutaron lo más exhaustivas posibles, puesto que para la muestra seleccionada (11 DCP) se pudieron encontrar mayor número de NC durante su ejecución, solamente se detectaron 6 NC.

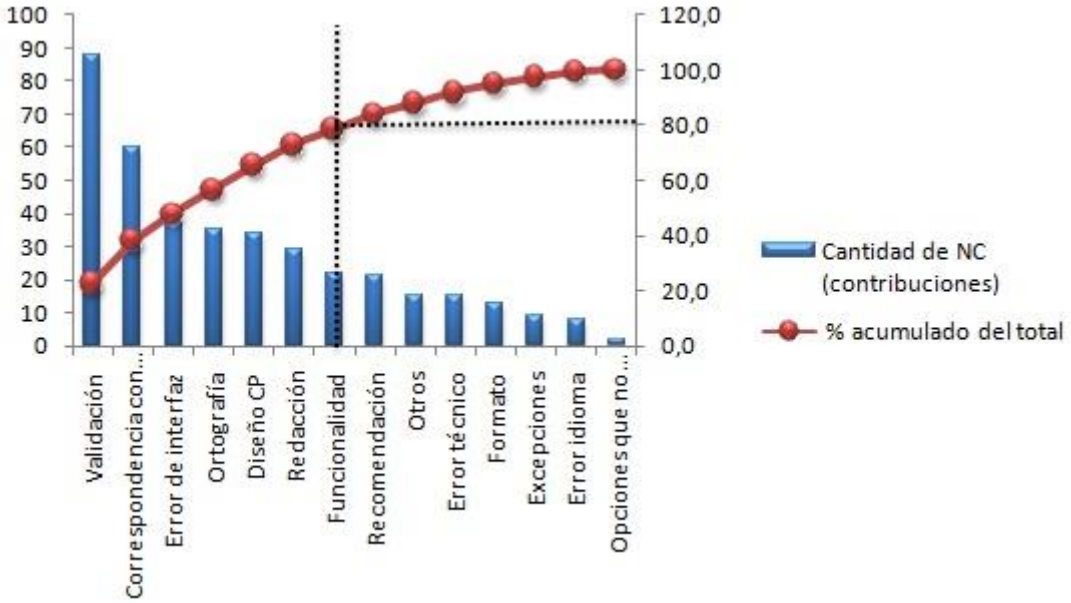
Es imprescindible que los especialistas de pruebas estén totalmente enfocados en la prueba que realizarán y tengan todas las condiciones necesarias para una eficiente ejecución.

Otro aspecto interesante en esta comparación, es la diferencia de los tipos de NC que se encuentran entre los artefactos que se le aplican las PE y los que no. Se evidencia que durante la ejecución de la primera iteración a los artefactos que no se le aplican las PE, las NC que se detectan son más técnicas, de mayor complejidad, en las gráficas que se muestran a continuación se evidencia la diferencia. Aplicando las gráficas de Pareto (en el anexo 12 se ejemplifican) se evidencia que en la primera gráfica (Gráfica 1) la cual representa la cantidad total de NC detectadas durante la primera iteración sin haberse ejecutado las PEI, el 20 % de las NC detectadas son de tipo funcionalidad, validación, redacción, error de interfaz y ortografía, aspectos técnicos principalmente.

A su vez en la segunda gráfica (Gráfica 2) que representa la cantidad total de NC detectadas durante la 1ra iteración después de haberse ejecutado las PEI, las NC detectadas son de tipo validación, correspondencia con otro artefacto, error de interfaz, ortografía, diseño de casos de prueba y redacción, aspectos menos técnicos, pero igualmente de gran importancia.



Gráfica 1: Cantidad total de NC detectadas durante la 1ra iteración de prueba sin haberse ejecutado las PEI.



Gráfica 2: Cantidad total de NC detectadas durante la 1ra iteración después de haberse ejecutado las PEI.

3.1.2. Tiempo de ejecución de las pruebas

El tiempo total de las pruebas que se tiene en cuenta en esta investigación, es el tiempo destinado a cada una de las actividades definidas en el ciclo de PE y el tiempo dedicado a ejecutar la primera iteración de prueba sin habérsele incluido la estrategia de PE.

Para demostrar como disminuye el tiempo en la ejecución de las pruebas a partir de la inserción de la estrategia de PE en el LIPS, se establece una comparación del tiempo que demoró la ejecución de la primera iteración entre artefactos que se le aplicaron las PEI y artefactos que no. Con el objetivo de obtener datos que permitieran demostrar esta variable se realizaron búsquedas en los repositorios del DPSW y se utilizó como método la entrevista a profundidad a especialistas del LIPS. Se seleccionaron 16 proyectos (a 8 se le aplicaron las PEI y los restantes no) que tuviesen la misma o similar cantidad de CP, funcionalidades, requisitos o páginas, presentaran complejidad equivalente y los mismos artefactos.

A continuación se presenta la Tabla 9 con el estudio comparativo, donde se precisa el tipo de artefacto, la cantidad de CP, funcionalidades, requisitos o páginas de los mismos, el tiempo dedicado a la 1ra iteración de pruebas con la aplicación de las PEI y sin dichas pruebas y finalmente el estado que se obtuvo de la evaluación de los artefactos durante las PEI.

Tabla 9: Comparación entre el tiempo de ejecución de las pruebas de proyectos que se le aplicaron PE y proyectos que no.

En las PEI					Sin ejecutar PE (1ra it)				Después de las PE (1ra it)				Resultados
Proyecto	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Tiempo (días ó horas)**	**Estado**	**Proyecto**	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Tiempo (días ó horas)**	**Proyecto**	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Tiempo (días ó horas)**	**Tiempo ahorrado**
A	Manual de usuario	28 páginas	2 horas	P. abortadas	B	Manual de usuario	16 páginas	5 días	A	Manual de usuario	28 páginas	1 día	**4 días**
C	Sistema	45 CP (11 CP de muestra)	2 días	P. satisfactorias	D	Sistema	64 CP	1 mes (30 días)	C	Sistema	45 CP	25 días	**5 días**
E	Sistema	9 CP (2 CP de muestra)	4 horas	P. abortadas	F	Sistema	14 CP	14 días	E	Sistema	9 CP	2 horas	**13 días**
G	Portal web	23 CP (5 CP muestra)	1 día	P. satisfactorias	H	Aplica-ción web	21 CP	2 meses (60 días)	G	Portal web	23 CP	4 días	**56 días**
I	Sistema	63 CP (16 CP de muestra)	4 días	P. abortadas	J	Sistema	61 CP	13 días	I	Sistema	63 CP	6 días	**7 días**
K	Sistema	73 CP (19 CP de muestra)	1 día	P. satisfactorias	L	Sistema	65 CP	36 días	K	Sistema	73 CP	30 días	**6 días**
M	Sistema	4 CP	4 horas	P. satisfactorias	N	Sistema	5 CP	2 días	M	Sistema	4 CP	1 día y 4 horas	**4 horas**
O	Sistema	7 CP (2 CP de muestra)	4 horas	P. satisfactorias	P	Sistema	9 CP	9 días	O	Sistema	7 CP	3 días	**6 días**

Como se puede apreciar en la última columna de la tabla anterior la cantidad de horas y días que disminuye el tiempo es significativo, lo que demuestra que con la aplicación de la estrategia de PE el tiempo dedicado a ejecutar la primera iteración de prueba es menor que durante una iteración completa sin las PEI. Esto se logra dado que en la nueva conceptualización del LIPS se ha introducido la propuesta plasmada en esta investigación de realizar un grupo de pruebas a una porción del artefacto a liberar en un período determinado de tiempo, antes de comenzar la primera iteración de prueba.

Es válido aclarar que en este caso existen un grupo de factores de riesgos a tener en cuenta, que no son responsabilidad de los especialistas en sí, pero que influyen en el tiempo de ejecución de las pruebas. Ejemplo de ello lo constituye: la asistencia de los estudiantes de 2do año al LIPS, la cantidad de proyectos que se le asignan a un mismo especialista, la existencia de proyectos priorizados con los cuales se detienen las demás pruebas en ejecución y el nivel de entrenamiento y superación de los especialistas en las pruebas.

De un total de 20 artefactos que se le ejecutaron las PEI los cuáles fueron tomados como muestra para establecer la comparación de tiempo y no conformidades detectadas, 7 de ellos sus pruebas fueron abortadas y 13 satisfactorias, lo que indica que éstos 7 artefactos abortados fueron devueltos a los proyectos, pues en su evaluación cumplían con los criterios de criticidad definidos por Calisoft, lo que evidencia la necesidad y efectividad de la aplicación de la estrategia de PE, de lo contrario se hubiese perdido tiempo, esfuerzo y recursos en ejecutar estas pruebas directamente en la primera iteración del proceso de liberación.

3.1.3. Optimización de recursos

Según la Real Academia de la Lengua Española (RAE) optimizar significa mejorar la manera de realizar una actividad. En este sentido en la estrategia se propone optimizar los recursos humanos con los que cuenta el laboratorio, a través de una mejor distribución y gestión de los especialistas de pruebas.

En esta investigación se propone la utilización de un grupo independiente de pruebas los cuales serían los especialistas del LIPS, proporcionando principalmente conocimiento especializado que la organización independiente tiene respecto a las pruebas.

Para potenciar la especialización de los especialistas en diferentes tipos de prueba es que en la nueva conceptualización del LIPS, propuesta desarrollada por la MCs Tayché Capote y puesta en práctica desde el curso pasado en el DPSW, se propone la utilización de estudiantes como probadores, una fuerza de trabajo permanente para la ejecución de las

pruebas, lo cual ha permitido una mejor distribución de los recursos humanos en el departamento. Esto permite que los especialistas del laboratorio puedan enfocarse en desarrollar nuevos tipos de prueba y herramientas automatizadas para su ejecución, ejemplo de esto lo constituyen las pruebas exploratorias.

El interés por utilizar especialistas independientes del proyecto es porque ellos presentan un grupo de cualidades necesarias que debe tener un probador; mayor motivación en el proceso de prueba, oposición de intereses con el equipo de desarrollo y separación del proceso de prueba del control gerencial del proyecto.

3.1.4. Esfuerzo dedicado a las pruebas

Una de las propuestas que se presenta en esta investigación es ir realizando el reporte del esfuerzo invertido en cada etapa durante la realización de las actividades. Al comienzo de cada etapa se estima el tiempo que lleva realizar cada una de las actividades de la etapa y luego que las actividades son realizadas, se reporta el tiempo insumido en las mismas. A continuación se presenta la Tabla 10 con un ejemplo donde se muestra el esfuerzo invertido en la etapa Ejecución de la prueba por parte del especialista de prueba, tomado del proyecto E expuesto en la Tabla 9, donde el tiempo total del ciclo de PE fueron 4 horas. En la actividad EJ3 - Ejecución de la prueba, se fusiona el esfuerzo de las actividades EJ1 - Utilización de la muestra seleccionada y EJ2 - Utilización de documentos de apoyo a la prueba, ya que estas actividades fueron realizadas en forma simultánea.

Tabla 10: Esfuerzo en la etapa de Ejecución de la prueba.

Etapa – Ejecución de la prueba (EJ)	
Actividades	**Horas por actividad - Especialista de prueba**
EJ3 - Ejecución de la prueba	2 horas
EJ3.1 – Recolección de datos para el cálculo de las métricas	10 minutos
EJ4 - Redacción y descripción de las no conformidades	30 minutos
EJ5 - Revisión y asignación de las no conformidades	10 minutos
EJ6 - Reporte de las no conformidades	10 minutos
Total	**3 horas**

De esta forma es calculado el esfuerzo en cada etapa de la estrategia y una vez culminado el ciclo de prueba se suman todos los valores de los esfuerzos por etapa dando como resultado el esfuerzo total invertido en ejecutar el ciclo de pruebas exploratorias.

No se puede realizar una comparación de esta actividad entre el proyecto que se le haya ejecutado PE y el que no, puesto que esta actividad no estaba incluida y por tanto no se realizaba durante las pruebas de liberación. Sin embargo, se puede tener un estimado de cuanto esfuerzo requirió realizar las pruebas de los proyectos que se muestran en la Tabla

9, dado que el esfuerzo se mide horas/hombre, y como se muestra en dicha tabla el tiempo que se le dedicó a ejecutar la primera iteración de prueba de los proyectos que no se le aplicaron las PE es mucho mayor que el tiempo dedicado a ejecutar las PE, por lo que se infiere que el esfuerzo durante la ejecución de las pruebas es menor.

Es importante tener en cuenta que existen un grupo de factores que influyen directamente en el tiempo de ejecución de las pruebas y que no son precisamente dificultades del especialista, sin embargo es tiempo que se le agrega al cálculo del esfuerzo. Estos riesgos una vez culminado todo el ciclo de PE se tienen en cuenta y se analizan para su posterior tratamiento. De ahí la importancia de realizar siempre el cálculo del esfuerzo dedicado a las pruebas, puesto que las mediciones realizadas durante el ciclo de prueba son la base para las estimaciones de los proyectos posteriores, el concepto es aprender de cada una de las pruebas que se ejecuten, para perfeccionar cada vez más el proceso de pruebas de liberación del DPSW.

3.1.5. Aplicación de las métricas propuestas

Utilizando la misma selección de los artefactos anteriores se le aplicarán las métricas de rendimiento y funcionalidad para medir el por ciento de cumplimiento de las características de calidad, evaluando el producto desde el mismo proceso de prueba y aportando más elementos para medir el rendimiento del laboratorio.

- Métrica de rendimiento

Para la selección de la media a comparar se utilizaron los datos proporcionados por la tesis “Catálogo automatizado de métricas de calidad para evaluar los productos en las pruebas”. El parámetro de comparación del resultado de la métrica de rendimiento aplicada a los proyectos es 2.1 y mientras mayor sea el resultado mejor es el por ciento de cumplimiento de la característica de eficiencia y por ende la sub-característica de rendimiento.

Tabla 11: Resultado de la métrica de eficiencia.

Sub-características de calidad	Nombre de la métrica	Métrica 1ra iteración – con PEI	Métrica 1ra iteración – sin PEI
Rendimiento	Rendimiento	4,5	1

Como se muestra en la tabla anterior el resultado de la métrica aplicada al proyecto que ejecutó PEI en la primera iteración supera la media establecida. Esto indica que con la estrategia de PE se puede probar mayor cantidad de CP en menor tiempo, no siendo así en el proyecto que no se le aplica las PE, como se muestra en la tabla anterior este presenta un valor menor que la media, por lo que demora mayor tiempo en ejecutar la misma cantidad de CP. Se puede concluir que con la aplicación de las PE mejora el rendimiento del laboratorio, puesto que disminuye el tiempo de ejecución de las pruebas.

- Métrica de funcionalidad

En este caso la comparación se realizaría entre el resultado de la propia métrica, la interpretación del valor indica que mientras el cálculo de la métrica se acerque a 1 el resultado es mejor.

Tabla 12: Resultado de la métrica de funcionalidad.

Sub-características de calidad	Nombre de la métrica	Métrica 1ra iteración – con PEI	Métrica 1ra iteración – sin PEI
Idoneidad	Adecuación funcional	0,6	0,3

Como se muestra en la tabla anterior el resultado obtenido de la métrica aplicada al proyecto que ejecutó PEI en la primera iteración tiene un valor que se acerca más a 1, lo que demuestra mayor adecuación funcional de la aplicación de los requisitos establecidos por el cliente. No siendo así en el proyecto que no se le aplica las PE, puesto que se detectan mayor cantidad de NC en igual cantidad de casos de prueba, demostrando que con la aplicación de las PE se logra mayor eficiencia en el proceso de pruebas liberación.

En el anexo 21 se muestra el valor que se le asignó a las variables de las métricas según la recolección que se obtuvo de los datos de los proyectos y los resultados del cálculo de las mismas.

3.1.6. Diagnóstico para validar la estrategia de pruebas exploratorias

3.1.6.1. Encuesta # 2. Evaluación de la estrategia de pruebas exploratorias

La población a estudiar fue el personal involucrado en las pruebas de los productos de software en el LIPS, dígase, los especialistas del departamento de pruebas y como unidad de estudio el proceso de pruebas de liberación en el DPSW. La selección se realizó con la técnica de muestreo no probabilística, muestreo intencional para poder obtener la mayor representatividad e información posible, de acuerdo con los intereses de la investigación que fue entrevistar la mayor cantidad posible de especialistas que tienen experiencia en las pruebas de software y que han tenido que enfrentar diferentes problemáticas durante la ejecución de las pruebas, las cuáles le han permitido ir madurando.

La encuesta cuenta con 5 preguntas aplicada a 20 especialistas de pruebas, lo cual representa aproximadamente un 61 % del total de especialistas pertenecientes al LIPS vinculados directamente a la ejecución de las pruebas de liberación, (en el anexo 22 se puede ver el diseño de la encuesta # 2). La misma se realizó con el objetivo de identificar las potencialidades que brinda la aplicación de la estrategia y el nivel de cumplimiento de la

situación problémica y el problema, validar la propuesta de solución y recopilar elementos a tener en cuenta en la solución.

3.1.6.1.1. Análisis de los resultados de la aplicación de la encuesta # 2

En la pregunta # 1 se les propuso asignar un rango de importancia, utilidad y necesidad entre el 1 y el 10 aplicando la escala: Muy Alta para los valores 10 y 9; Alta para los valores 8 y 7; Media para los valores 6 y 5; Baja para los valores 4 y 3 y Ninguna para los valores 2 y 1.

Los resultados derivados de esta pregunta en cuanto al grado de **importancia y utilidad** que representa la estrategia de pruebas exploratorias para el DPSW, según el criterio de los especialistas de pruebas que se evalúan le otorgaron en el rango de Muy Alta y Alta un valor de 95% a cada uno de los indicadores. En el caso del grado de **necesidad** los encuestados expresan que para el LIPS dicho indicador en el rango de Muy Alta y Alta representa un 100%. Estas valoraciones reflejan la aprobación de los especialistas de prueba ante la estrategia presentada en la investigación y las potencialidades que brinda su aplicación para el DPSW. A continuación se muestran graficados los aspectos mencionados anteriormente.

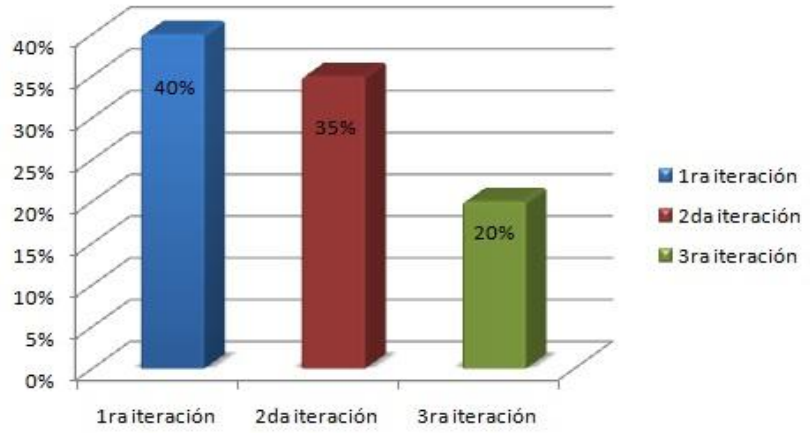
Gráfica 3: Grado de importancia de la estrategia de pruebas exploratorias.

Gráfica 4: Grado de utilidad de la estrategia de pruebas exploratorias.

Gráfica 5: Grado de necesidad de la estrategia de pruebas exploratorias.

Los resultados obtenidos de las preguntas 2 y 3 fueron muy favorables, puesto que los encuestados expresan que con la aplicación de la estrategia de pruebas exploratorias en el LIPS se ahorra: tiempo en un 85%, esfuerzo en un 95% y recursos en un 90%, demostrando así que se hace más eficiente la ejecución de las pruebas, aumentando a su vez el rendimiento del laboratorio, valorado su cumplimiento por los especialistas en un 90%.

En la investigación se propone que además de realizar las pruebas exploratorias ante la primera iteración de pruebas éstas puedan ser aplicadas paralelas a la segunda iteración. Sin embargo, según el criterio de los especialistas encuestados expresan que por la forma en que está conformada la estrategia y los beneficios que esta aporta, puede ser aplicada también en las demás iteraciones de prueba, en la gráfica que se muestra a continuación se puede observar la medida en que se propone ejecutar las PE durante las iteraciones de prueba.



Gráfica 6: Inserción de las pruebas exploratorias en las diferentes iteraciones de pruebas.

Una de las facilidades que brinda la estrategia y que se propone en la investigación es su posible aplicación en cualquier laboratorio de pruebas de software, al realizarle esta pregunta a los encuestados el 100% estuvo de acuerdo con su aplicación, fundamentando que es factible, pues la estrategia utiliza pocos recursos (lo expresan así el 80%), porque

durante su ejecución no es necesario utilizar pruebas especializadas (lo expresan así el 40%), porque el tiempo empleado en su ejecución es menor que el de una iteración completa de pruebas (lo expresan así el 95%), porque el cliente puede obtener una evaluación de la calidad del producto en breve plazo (lo expresan así el 55%) y porque se puede ejecutar con pequeños equipos (lo expresan así el 30%).

Los resultados obtenidos con la encuesta son satisfactorios, para la mayoría de los encuestados la estrategia propuesta les parece coherente con las necesidades planteadas en la investigación, de mucha actualidad e interesante, con posibilidades de aplicación en laboratorios de pruebas nacionales e internacionales, dado que aborda temas de interés y propone soluciones para cualquier organización que se dedique a realizar pruebas de software. Expresan que la estrategia puede contribuir a que los laboratorios mejoren su rendimiento, y por ende sean más eficiente, objetivo fundamental que fue planteado en la investigación y que su mayor aporte es práctico sobre todo para el DPSW donde no se utilizaba esta técnica como recurso para lograr mayor eficiencia durante la ejecución de las pruebas de liberación.

Los encuestados recomiendan profundizar en un grupo de aspectos para potenciar la estrategia, ellos son:

- Se deben definir criterios que indiquen la necesidad de realizar una prueba exploratoria.
- Se debe tener en cuenta el control para potenciar los indicadores de rendimiento y eficiencia.
- Evaluar la posibilidad de lograr mayor especialización de las pruebas exploratorias cuando son ejecutadas en paralelo a las diferentes iteraciones de pruebas.

3.2. Conclusiones parciales del capítulo III

- Se implantó la estrategia de pruebas exploratorias en el DPSW.
- Se realizó un análisis comparativo entre la cantidad de NC detectadas durante la ejecución de las PEI y la primera iteración, demostrando un aumento en la calidad de las pruebas, puesto que disminuyen con la aplicación de las PE la cantidad de NC que se detectan.
- Se demostró cómo disminuye el tiempo en la ejecución de las pruebas mediante un análisis comparativo entre el tiempo de ejecución de las pruebas durante la primera iteración con la aplicación de las PEI y sin su ejecución.
- Se aplicaron métricas propuestas en la estrategia a través de verificaciones realizadas durante las pruebas a proyectos seleccionados, logrando medidas de la efectividad del proceso de prueba a partir de la evaluación de los productos liberados, demostrando cómo mejora el rendimiento del laboratorio mediante la disminución del tiempo de

ejecución de las pruebas y aumento de la eficiencia en el proceso de pruebas de liberación a través de la detección de mayor cantidad de NC en similares cantidades de casos de pruebas.

- Se analizaron los resultados que se obtuvo de la aplicación de los métodos científicos mediante una encuesta, donde se pudo demostrar las potencialidades que brinda la aplicación de la estrategia y el nivel de cumplimiento de la situación problémica y el problema, validando de esta forma la propuesta de solución y recopilando los elementos a tener en cuenta en la solución.
- En esta investigación se enfoca el esfuerzo en las contribuciones más importantes, funcionalidades más propensas a fallar, y en el esfuerzo dedicado a las pruebas, con el objetivo de disminuir el tiempo, optimizar recursos, para obtener mayor beneficio (calidad en las pruebas) durante la ejecución de las pruebas y por ende según el análisis realizado de las NC detectadas durante las PE mejorar el rendimiento del LIPS.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Se creó la estrategia de pruebas exploratorias a partir de introducir elementos novedosos como la selección de la muestra en el proceso del LIPS.
- Se cumplió con el objetivo planteado en la investigación al implantar la estrategia de PE en el LIPS.
- Con la aplicación práctica de la estrategia se demostró la hipótesis de la investigación, puesto que se logró mayor eficiencia durante la ejecución de las pruebas. Constituyendo a su vez una técnica para mejorar el rendimiento del LIPS. Su integración con otros tipos de pruebas, con diferentes estrategias y procedimientos definidos por especialistas del LIPS contribuye a perfeccionar y fortalecer la calidad con que se ejecutan las pruebas en el laboratorio.

Recomendaciones

La estrategia propuesta en esta investigación ha sido incorporada en la nueva conceptualización del LIPS, su implantación ha permitido mejorar el rendimiento del laboratorio la cual muestra sus primeros resultados, sin embargo, los objetivos del trabajo no abarcan todos los elementos a definir; los cuales son amplios y diversos. Por lo que se propone:

- Aplicar la estrategia de pruebas exploratorias en las Pruebas de Aceptación.
- Profundizar en la selección de la muestra, específicamente en métodos de los elementos a probar y aspectos arquitectónicamente más significativos.
- Obtener las métricas propuestas en la estrategia que no se llegaron a calcular, con el objetivo de evaluar el proceso y los productos en las pruebas, para así tener más elementos a la hora de emitir una evaluación del proceso y los productos, según el cumplimiento de las características de calidad de la ISO 9126-1 Parte 1: Modelo de calidad y las métricas adaptadas de la literatura estudiada.
- Aplicar el Diagrama de Pareto para mostrar la comparación de las NC de un mismo artefacto en las diferentes iteraciones, lo cual le permite al LIPS visualizar el avance que va teniendo el artefacto y poder gestionar mejor los recursos en función de los resultados de las pruebas por iteraciones. Además esta herramienta sirve como medidor para determinar si se mantiene o no la estrategia de prueba que se lleva a cabo.
- Aplicar, evaluar y revisar la implantación de la estrategia en las futuras sedes de Calisoft diseminadas por todo el país.

REFERENCIAS BIBLIOGRÁFICAS

AEC, A. E. d. C. (2009). *"Anuario Estadístico de Cuba 2009"*. Oficina Nacional de Estadística, disponible en: http://www.one.cu/aec2009.htm.

Álvarez. S, F. A., Fernández. H. (2000). *"Aplicación del modelo CMM a la empresa Segurmática".* Ingeniería Industrial, ISPJAE.

Apache Software Foundation. (1999-2010). *"The Apache Jakarta Project"*. Disponible en: http://jakarta.apache.org/jmeter/.

Bach, J. (2000). "*General Functionality and Stability Test Procedure*". Desktop Applications Edition. Disponible en: www.satisfice.com.

Bach, J. (2002). *"Exploratory Testing Explained"*. The Test Practicioner. Disponible en: http://www.satisfice.com/articles/et-article.pdf.

Bach, J. (2002, noviembre). *"Session Based Test Management"*. Software Testing and Quality Engineering. 2.

Badle, S. K., Shinya, Comité Selenium. (2010). *"SeleniumHQ, web application testing system"*. Disponible en: http://seleniumhq.org/.

Basili V., C. G., Rombach D. (1994) *"The Goal Question Metric Approach"*. Wiley& Sons Inc.

BASTOS, T. P. S. M., Felipe. (2009). *"Desafíos y oportunidades de la industria del software en América Latina"*. B. S. M. Teresa. Colombia, Cepal en coedición con Mayol Ediciones S.A.

Beck, K. (1999). *"Extreme Programming Explained, Embrace Change"*. A. Wesley. 201-61641-6.

Besterfield, D. (2010). *"Control de la Calidad"*, 1ra edición.

Black, R. (2002). *"Managing the Testing Process",* 2da edition, Editorial Wiley, 0-471-22398-0.

Boehm, B. (1984). *"Verifying and Validating Software Requirements and Design Specifications"*. IEEE Software. 1: 75-88.

Boston, M. (2009). *"New Standish Group report shows more project failing and less successful projects".* The Standish Group International, Inc.

Brualla, C. R. (2005). *"Curso de introducción a la Calidad y Mejora del Proceso Software"*. Disponible en: http://www.calidaddelsoftware.com/modules.php?name=News&file=article&sid=97.

Capote, T. (2011). *"Conceptualización e implantación de un Laboratorio Industrial de Pruebas de Software"*. Departamento de pruebas de software. Ciudad de la Habana, Universidad de las Ciencias Informáticas.

CaproSOFT. (2003). *"Programa de apoyo a la competitividad del sector de software"*. C. PROCOMER, CENAT.

CES. (1989). "*Centro de Ensayos de Software, CES"* , disponible en: http://www.ces.com.uy/.

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

Concyteg. *"Concyteg, contigo vamos"*. Disponible en: http://www.concyteg.gob.mx. Fecha de consulta: junio de 2011.

Choucair Testing SA. *"Choucair, Effective Software Testing"*. Disponible en: http://www.choucairtesting.com/. Fecha de consulta: junio de 2011.

Daich, G., Price G., Raglund B., Dawood M. (1994) *"Software Test Technologies Report"*. Software Technology Support Center.

Dijkstra, E. (1970). *"Notes on structures Programming"*. TH Report 70 –WSK-03. Disponible en: http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.pdf.

Domínguez, J. (2009) *"The curious case of the CHAOS Report 2009"* .Project Smart.co.uk. Disponible en: http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-20.

Dustin, E., Rashka, Jeff., Paul, John. (2008). *"Automated Software Testing: introduction, management and performance"*. New York y Canadá, United States at Hamilton in Castleton. Simultáneamente en Canadá.

El Economista de Cuba. (2009) *"India: La ola que viene, crece la industria india de tecnología de información y de servicios remotos"*. Edición Online, El economista de Cuba. Digital de la Asociación Nacional de Economistas y Contadores de Cuba 2009. Disponible en: http://www.eleconomista.cubaweb.cu/index.html.

Equallity.com. *"Equallity.com. Lo que usted necesita, cuando lo necesita". Disponible en:* http://www.equallity.com/. Fecha de consulta: junio de 2011.

Febles Estrada, A. (2003). *"Un modelo de Referencia para la Gestión de Configuración en la PYME de Software".* Ciudad de la Habana, CUJAE.

Fernández, L. (2001). *"Calidad del Software"*. ATI (Asociación de Técnicos de Informática). España. Disponible en: http://www.ati.es/gt/calidad-software/presentacion.htm.

Fernández, P. J. M. (2002, marzo). *"Pruebas de integración para componentes de software"*. Instituto Politécnico Nacional Centro de Investigación en Computación. Tesis para obtener el grado de Doctor en Ciencias de la Computación.

Ferris, T. (2010). "*La semana laboral de 4 horas"*, 1ra edición.

Geeknet, I. (2011). *"Source Force. DBmonster, feed your database"*. Disponible en: http://sourceforge.net/projects/dbmonster/.

Gómez, A. L. (2011). *"Estrategia de calidad-PARQUESOFT"*. Fundación Parque Tecnológico del Software. Disponible en: http://www.buenastareas.com/ensayos/Estrategia-De-Calidad-Parquesoft-Colombia/2696155.html.

Góngora, A. (2012). *"Catálogo automatizado de métricas de calidad para evaluar los productos en las pruebas"*. Departamento de Pruebas de Software. Ciudad de la Habana, Universidad de las Ciencias Informáticas: 111.

González, B. D. L., .Rodenes, Adam Manuel,. (2007). *"Factores Críticos de éxito de la Industria de Software y su relación con la orientación estratégica de negocio: un estudio empírico exploratorio".* Gestión de Tecnología y Sistemas de Información 4(1): 47-70.

Hernández León, R. A. C. G. S. (2002). *"El paradigma cuantitativo de la investigación científica"*. EDUNIV, pág. 114, 959-16-0343-6.

Hetzel, B. (1988). *"The complete guide to Software Testing, 2nd Edition"*, QED Information Sciences Inc.

Hexaware Technologies. *"Hexaware Technologies, your success is our focus"*. Disponible en: http://www.hexaware.com/. Fecha de consulta: junio de 2011.

Hexaware Technologies. (2011). *"Hexaware sets up HP Software Lab in Mexico Strengthens association with HP Software"*. Disponible en: http://www.hexaware.com/fileadd/I875_Press_Release_%20Hexaware%20sets%20up%20HP%20Software%20Lab%20in%20Mexico.pdf

IEEE, (1990). *"IEEE-610-12-Standard Glossary of Software Engineering Terminology Institute of Electrical and Electronics Engineers"*. ISBN: 155937067X. Disponible en: http://www.ieee.org/sitemap.html. Fecha de consulta: enero de 2011.

IEEE, (2002). *"IEEE-Std-610-12-Standard Glossary of Software Engineering Terminology"*. Disponible en: http://www.ieee.org/sitemap.html. Fecha de consulta: enero de 2011.

ISO, (2000). "*ISO 9000:2000-Sistemas de gestión de la calidad-Fundamentos y vocabulario"*. Disponible en: http://www.iso.org/iso/home.html. Fecha de consulta: enero de 2011.

ISTQB. (2005). *"International Software Testing Qualifications Board, Certified Tester Foundation Level Syllabus"*. Disponible en: http://www.istqb.org/fileadmin/media/SyllabusFoundation.pdf.

Jacobson, I., Booch G., Rumbaugh J. (1999). *"The Unified Software Development Process"*, Addison Wesley.

Kaner C. (2001). *"Measurement Issues and Software Testing"* . QUEST conference, Orlando FL. Disponible en: http://www.kaner.com/pdfs/measurement_segue.pdf.

Kaner C., B. J., Pretichord B. (2001). *"Lessons Learned in Software Testing"*. ISBN 0471081124. Disponible en: http://www.istqb.org/fileadmin/media/SyllabusFoundation.pdf.

Kaner C., F. J., Nguyen H. (1999). *"Testing Computer Software"*, 2da edition. ISBN: 0471358460.

Kit, E. (1995). *"Software Testing In The Real World : Improving The Process"*. Addison Wesley. ISBN 0201877562.

Lagares. P, J. P. (2001). *"Población y muestra. Técnicas de muestreo"*. MaMaEuSch. Sevilla, Management Mathematics for European Schools.

Lage, C. (2000). *"Discurso pronunciado en la Inauguración de la Convención Informática 2000"*. La Habana, Cuba, Informática 2000.

Lamancha, P. B. (2007). *"Gestión de las Pruebas Funcionales"*. Actas de Talleres de Ingeniería del Software y Bases de Datos, . Montevideo, Uruguay, Universidad de la República. Vol.1.

Lang, J. P. (2006-2011). *"Redmine*", disponible en: http://www.redmine.org/.

LNCS. *"Instituto Nacional de Tecnologías de la Comunicación, INTECO"*. Disponible en: http://www.inteco.es/. Fecha de consulta: junio de 2011.

Lovelle, C. J. M. (1999). *"Calidad del Software"*, Universidad Nacional de la Pampa, Grupo GIDIS. Disponible en: http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.pdf.

McCall, J., Richards, P., & Waters, G. (1977). *"Factors in Software Quality"*, Rome Air Development Center. RADC-TR-77-369.

Menéndez, H. (2012). "*Procedimiento para el diseño e implantación de pruebas a partir del análisis de los riesgos del producto".* Departamento de Pruebas de Software. Ciudad de la Habana, Universidad de las Ciencias Informáticas: 88.

Myers G. (2004). *"The art of software testing"*. John Wiley & Sons Inc. ISBN 0-471-46912-2.

ISO, (2005). *"NC-ISO-IEC-9126-1-Ingeniería de software-Calidad del producto-Parte 1: Modelo de calidad"*. Oficina Nacional de Normalización, ONN. Disponible en: http://www.iso.org/iso/home.html.

ISO, (2005). *"NC-ISO-IEC-9126-2-Ingeniería de software-Calidad del producto-Parte 2: Métricas externas"*. Oficina Nacional de Normalización, ONN. Disponible en: http://www.iso.org/iso/home.html.

ParqueSoft. (2009). *"ParqueSoft, 10 años haciendo"*. Disponible en: http://www.parquesoft.com/.

Pérez , B. P., A; Travieso, M; Wodzislawski M. (2007). *"Testing exploratorio en la práctica". Disponible en: http://www.ces.com.uy/documentos/JIISIC-2007.pdf.*

Pérez Lamancha, B. (2006). *"Proceso de Testing Funcional Independiente"*. Instituto de Computación. Uruguay, Universidad de la República: 164.

Pointe Technology Group, I. *"Software Testing and Quality Assurance White Papers"*. Quality Technology Solutions. Disponible en: www.pointetech.com. Fecha de consulta: junio de 2011.

Pomeroy-Huff M., M. J., Cannon R., Sebern S. (2005). *"The Personal Software Process (PSP) Body of Knowledge"*. Versión 1.0, CMU/SEI-2005-SR-2003. Disponible en: http://www.sei.cmu.edu/publications/documents/05.reports/05sr003.html.

Pressman, R. S. (2005). *"Ingeniería de Software, un enfoque práctico".* 6ta. edición, Mc Graw Hill. ISBN 970–10–5473–3.

Pressman, R. S. (2005a). *"Ingeniería de software: un enfoque práctico"*. 6ta edición. No. de páginas 614, 970–10–5473–3.

Pressman, R. S. (2002). *"Ingeniería del Software. Un enfoque práctico"*. Mc Graw Hill.

Robinson, H. *"Exploratory Modeling"*. Disponible en: http://www.testingcraft.com/exploratory-robinson.html. Fecha de consulta: enero de 2011.

Rojas, J. B., Emilio. (2007). *"Herramientas para el seguimiento de defectos"*. Disponible en: http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node67.html.

Rovira, C. *"Diagrama de Pareto. Herramienta básica para la mejora de la calidad".* OP Group. Argentina. Disponible en: http: //www.op.-group.net/. Fecha de consulta: enero de 2011.

Santos Hernández, V. (2009). *"La industria del software. Estudio a nivel global y América Latina"*. Revista académica de economía. Observatorio de la Economía Latinoamericana. (116), ISSN 1696-8352. Disponible en: http://www.eumed.net/cursecon/ecolat/la/09/vsh.htm.

Scalone, F. (2006). *"Estudio comparativo de los modelos y estándares de calidad de software".* Maestría en Ingeniería en Calidad. Facultad Regional de Buenos Aires. Buenos Aires, Universidad Tecnológica Nacional: 461.

SIME. (1997). *"Lineamientos estratégicos para la informatización de la sociedad cubana".* La Habana, Cuba.

Sogeti. (2009). *"Sogeti, Local touch-global each"*. Disponible en: http://www.es.sogeti.com/.

SWEBOK. (2004). *"Guide to the Software Engineering Body of Knowledge SWEBOK"*. versión 2004. IEEE Computer Society. Disponible en: http://www.swebok.org.

Valdivia, D. R. V. E. E. G. E. (2005). *"Estándares de calidad para pruebas de software"*. Sistemas e Informática, Universidad Nacional Mayor de San Marcos.

VMware, I. (2011). *"Temporary Maintenance"*. Disponible en: http://www.vmware.com/webmaint/maintenance.html.

VMware, I. (2011a). *"VMware vSphere Hypervisor Based on ESXi".* Disponible en: https://www.vmware.com/tryvmware/index.php?p=free-esxi&lp=1&biw=1024&bih=551&q=vsphere%20client%20download&revid=1221638444&sa=X&ei=AG3ETaXkHsPr0QHZoIijCA&ved=0CIIBENUCKAA.

Wasesores. (2010-2011). *"Wasesores. Meld, comparador de diferencias entre archivos".* Disponible en: http://wasesores.com/meld-diff-comparador-carpetas-archivos-ubuntu/.

Whittaker, J. (2000). *"What is Software Testing? And Why Is It So Hard?"*, IEEE Software. Volúmen 17, No. 1, pág 70-79.

BIBLIOGRAFÍA

Barreiro L. P y Albandoz P. J. (2001). *"Población y muestra. Técnicas de muestreo"*. Management Mathematics for European Schools, MaMaEuSch. 94342 - CP - 1 - DE - COMENIUS - C21, Universidad de Sevilla, España.

Black, R. (2002). "*Managing the Testing Process*", 2nd Edition, John Wiley & Sons.

Calvarro, N. J. (2009). *"Testing, porqué, cómo, cuándo y dónde"*. Software Quality Testing.

CEPAL. (2010). "*Estadísticas Económicas*". Anuario estadístico de América Latina y el Caribe.

Drummond, H. (1997). *"Qué es hoy la Calidad Total".* El movimiento de la Calidad. Deusto. Bilbao, España.

Feigenbaum, Armand V. (1991). *"Total Quality Control".* 3ª Edition, McGraw-Hill. New York. USA.

Ferreiro, J. R. (2010). *"La Mejora del Proceso Software".* ESI. Disponible en: http://www.aetic.es/CLI_AETIC/ftpportalweb/documentos/02Mejora%20de%20los%20Procesos%20del%20SW%20%5BModo%20de%20compatibilidad%5D.pdf.

Fernández, C. M. (2007). *"Sistemas de Calidad para Tecnologías de la Información".* Product Manager Certificación TICs, CISA, CISM. Disponible en: www.compite.org.mx/.../CY07-AENOR-Compite-%20TICs-VF-Sep-%202007.ppt.

Garzás, J. (2010). *"Tendencias actuales en la certificación e implantación de modelos y normas de calidad de software".* I Jornada de la calidad en los sistemas de información CSI. Disponible en: www.kybeleconsulting.com.

Fillottrani, P. R. (2007). *"Calidad en el Desarrollo de Software"*. Depto. Ciencias e Ingeniería de la Computación. Bahía Blanca, Universidad Nacional del Sur.

IEEE, (2003). IEEE Software Engineering Standards Collection, ISBN 978–0738137575. Inst of Elect & Electronic, 2003. Referenciado en 137. Disponible en: http://www.ieee.org/sitemap.html.

Kitchenham, B. y Pflegeer, S.L. (1996). *"Software Quality: The Elusive Target"*. IEEE Software 20(1): 12-21.

Lamancha, P. B. (2007). "*Gestión de las Pruebas Funcionales"*. Actas de Talleres de Ingeniería del Software y Bases de Datos. Montevideo, Uruguay, Universidad de la República. Vol.1.

Lamancha, P. B. (2007, diciembre). *"Estrategia de gestión de las pruebas funcionales en el Centro de Ensayos de Software".* REICIS Revista Española de Innovación, Calidad e Ingeniería del Software. Vol. 3, número 003, ISSN (Versión en línea): 1885-4486. Asociación de Técnicos de Informática. Madrid, España, pp. 28-41.

Mahapatra R. P. y Singh J. (2008*). "Improving the Effectiveness of Software Testing through Test Case Reduction"*. World Academy of Science, Engineering and Technology 37.

Maibaum, T. y Wassyng, A. (2008). "*A product focused approach to software certificacion".* Computer Volúmen: 41, Issue: 2: 91-93.

Marrero, A. I. (2008). *"La inteligencia de negocio desde la perspectiva cubana: retos y tendencias"*. Consultoría Biomundi/IDICT. Playa, Ciudad de la Habana, Cuba, Universidad de la Habana.

Melchor S. S y Parra G. J. (2010). *"Una revisión y comparativa de modelos de procesos de prueba"*. Escuela Superior de Ingeniería Informática, Universidad Rey Juan Carlos.

Merchan, L. (2007). *"Estudio de Factores Críticos de éxito local e Internacional para Empresas de la Industria de Software"* Universidad de San Buenaventura, Colombia.

Monje, M. R. *"Calidad del producto de software. De la teoría a la práctica".* I Jornada CSI Calidad en los Sistemas de Información. Fecha de consulta: marzo de 2011.

Navarrete, E. J. (2009). *"Conociendo India para iniciativas de negocios y la aplicación de las normas del Acuerdo Alcance Parcial Chile-India".* Guía País India, elaborada por la oficina Económica y Comercial de España en Nueva Delhi. Disponible en: http://www.indiga.org/bha/geo.htm.

Pérez, B. (2006), *"Proceso de Testing Funcional Independiente (ProTest)"*. Tesis de Maestría en Informática, PEDECIBA Informática, Facultad de Ingeniería, Universidad de la República, Uruguay.

Pérez, B., Pittier, A., Travieso, M. y Wodzislawski, M. (2007), *"Testing Exploratorio en la Práctica"*, VI Jornadas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC), Lima, Perú, pp. 43-49.

Santos Hernández, V. (2009), *"La industria del software. Estudio a nivel global y América Latina"* en Observatorio de la Economía Latinoamericana, Nº 116. Texto completo en http://www.eumed.net/cursecon/ecolat/la/09/vsh.htm.

Stephen H. Kan. (2002). *"Metrics and models in software quality engineering".* Second edition, ISBN 0–201–72915–6. Adisson wesley, 95–456. Referenciado en 126, 128, 135.

Valls Figueroa, W. (2006). *"Procedimiento para la Evaluación y Análisis de la Calidad en Destinos Turísticos de Sol y Playa".* Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas. ISPJAE, Ciudad de la Habana, Cuba.

ANEXOS

Anexo 1. Evaluación de proyectos según CHAOS Reports-Standish Group

Tabla 13: CHAOS Reports-Standish Group. IndSW - Evaluación de proyectos.

Evaluación	1994	1996	1998	2000	2002	2004	2006	2009
Exitosos	16%	27%	26%	28%	34%	29%	35%	32%
No Satisfactorio	53%	33%	46%	49%	51%	53%	46%	44%
Cancelados	31%	40%	28%	23%	15%	18%	19%	24%

Anexo 2. Ingresos por comercio de TICs desde el 2007 hasta el 2009, por conceptos de producción de software, servicios informáticos, paquetes y aplicaciones

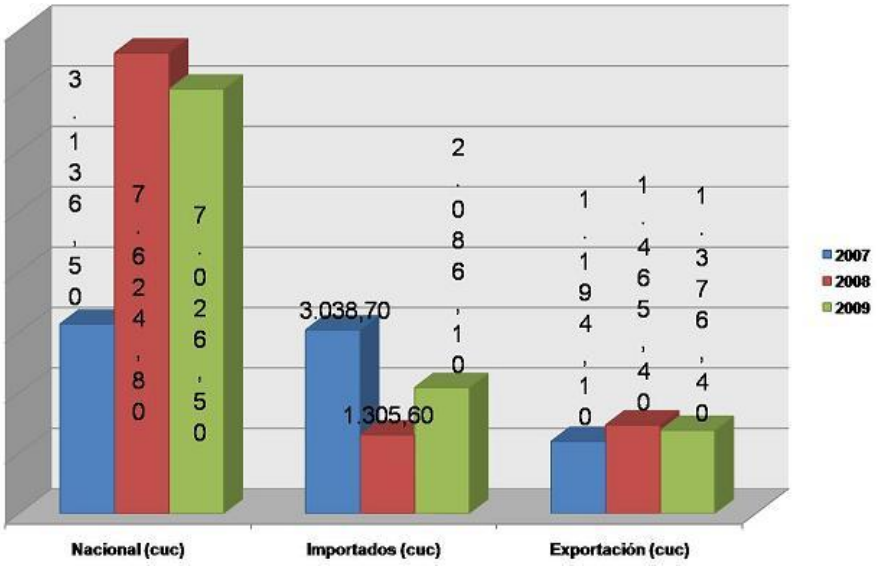


Figura 10: Representación gráfica de los ingresos por comercio de TICs - Producción de software.

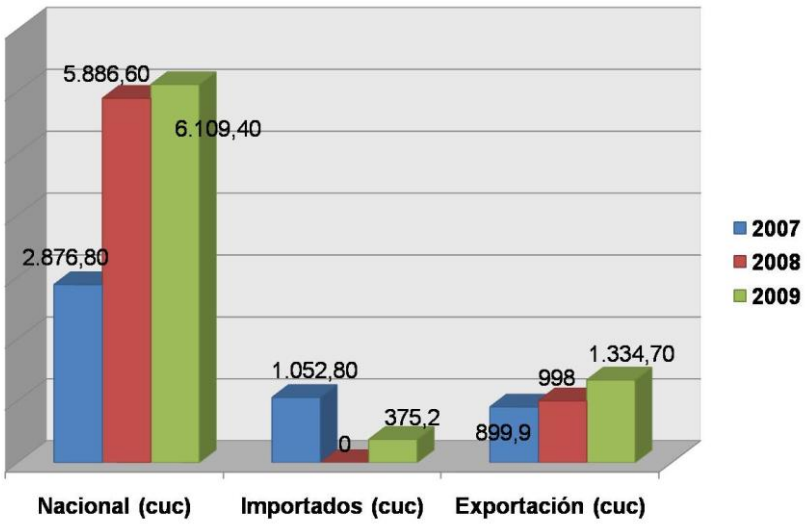


Figura 11: Representación gráfica de los ingresos por comercio de TICs – Servicios informáticos.

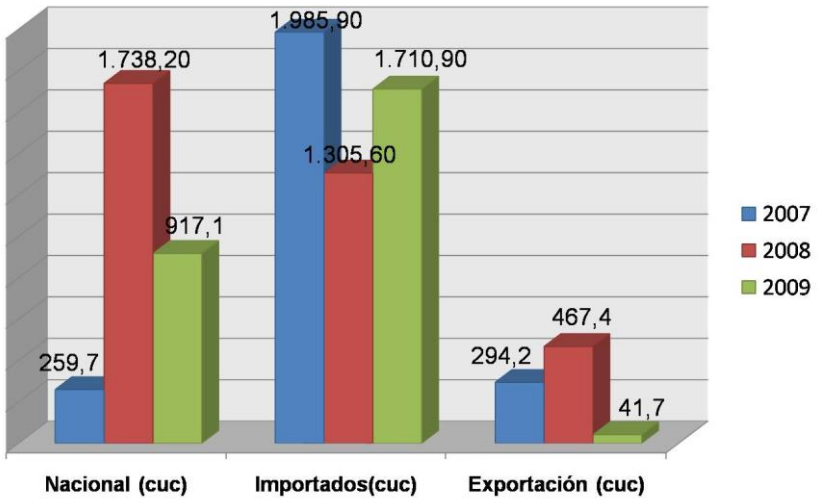


Figura 12: Representación gráfica de los ingresos por comercio de TICs - Paquetes y aplicaciones.

Anexo 3. Encuesta # 1 aplicada a los centros de desarrollo de la UCI

Datos Generales del Proyecto Productivo

Nombre del Centro	
Nombre del Proyecto	
Facultad	
Año de creación	
Tipo de proyecto	
Cantidad de Personas por proyecto	
Tiempo promedio de duración de los proyectos	

Cuestionario. Calidad y Pruebas de Software

Que categoría (categorías) responden a su actual posición:

- ❑ Directivo a nivel de centro
- ❑ Jefe de proyectos
- ❑ Jefe de proyecto o de equipo
- ❑ Desarrollador
- ❑ Otros

Otras actividades adicionales que Ud. Realiza:

- ❑ Control de los requerimientos del sistema
- ❑ Diseño del sistema
- ❑ Pruebas
- ❑ Aseguramiento de la calidad
- ❑ Control de la calidad
- ❑ Control de configuración
- ❑ Control de procesos
- ❑ Otros

Experiencia:

En su organización _____ años

En la producción de software _____ años

Su respuesta anónima sobre los siguientes aspectos nos será muy valiosa para medir el estado actual de la Calidad y las Pruebas de Software en la UCI. Marca con una cruz la evaluación que consideres correcta para cada parámetro, expresa tu opinión en el caso que lo consideres necesario.

---

1. Se dedica a:
   ❑ Producción de software
   □ Ciclo básico
   □ Ciclo profesional
   □ Investiga vinculado a una maestría
   □ Investiga vinculado a un doctorado
   □ Actividades de extensión (deportistas, artistas, etc.)

2. ¿Cómo catalogaría su conocimiento acerca de la Calidad del Software?
   ❑ Mucho □ Regular □ Poco □ Nada

3. Según su criterio la Calidad de Software se puede definir como:
   □ Medible y varía de un sistema a otro.
   □ Sinónimo de eficiencia, flexibilidad, corrección y confiabilidad.
   □ El grado en el que un conjunto de características inherentes cumplen con los requisitos.
   □ La concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos.
   □ Adecuación del producto al uso. Conformidad con requisitos y confiabilidad en el funcionamiento.

4. ¿Su proyecto tiene definido un cronograma de desarrollo?
   ❑ Sí □ No
   a) En caso afirmativo, están incluidas en el cronograma la ejecución de las pruebas de software:
      ❑ Sí □ No
   b) ¿Cuáles de las siguientes pruebas usted aplica en su proyecto?

□ Unidad
□ Integración
□Revisiones Técnicas Formales
□ Liberación
□ Validación
□ Sistema
□ Revisión Interna
□Regresión

c) ¿Con qué frecuencia su proyecto realiza pruebas?
□ Al finalizar el desarrollo de los componentes, subsistema o producto.
□ Antes de la entrega al cliente.
□ Antes de la implantación.
□ En el pilotaje.
□ Durante todo el ciclo de desarrollo de los componentes, subsistema o producto.
□ Cada dos semanas.
□ Durante la integración de los componentes, subsistema o producto.

5. ¿Cómo ve Ud. La aplicación de las Pruebas de Software en su proyecto?
   ❑ Adecuada □ Media □ Insuficiente

6. En caso de que considere que sea media o insuficiente dicha aplicación, ¿cuáles son, a su entender, las causas por la que esto ocurre?
   ❑ Falta de una adecuada estructura organizativa en el proyecto para este efecto.
   ❑ Desconocimiento del tema o técnicas a aplicar.
   ❑ Escaso personal capacitado y disponible para ello.
   ❑ Otros: _____

7. ¿Tiene su proyecto un equipo dedicado al Aseguramiento de la Calidad (SQA)?
   ❑ Sí □ No □No sé

8. Las pruebas que se le realizan al software según su criterio son:
   ❑ Muy importantes.
   ❑ Importantes pero no necesarias.
   ❑ Críticas.
   ❑ Crítica su realización para el desarrollo.
   ❑ Necesarias pero no importantes.
   ❑ Tan importantes como necesarias.

9. ¿Según los tipos de pruebas que se mencionan a continuación, tiene conocimiento cuáles se le han realizado a su componente, subsistema o producto?
   ❑ Funcionalidad.
   ❑ Confiabilidad.
   ❑ Usabilidad.
   ❑ Eficiencia.
   ❑ Mantenibilidad.
   ❑ Portabilidad.
   ❑ Fiabilidad.
   ❑ Revisión Técnica a la documentación.
   ❑ Otras, diga cuál: _____

10. ¿Existe en su proyecto un procedimiento, proceso o metodología que guíe las pruebas y sea seguida durante el desarrollo del mismo?
    ❑ Sí □ No □No sé

11. Las Pruebas Exploratorias ó Testing Exploratorio se definen como: el proceso simultáneo de exploración del producto (aprendizaje), diseño y ejecución de pruebas.
    ¿Aplica su proyecto este tipo de Pruebas?

- ❑ Sí □ No □No sé

a) En caso de ser su respuesta afirmativa, mencione en qué momento se aplica:
   - ❑ Cuándo no se conoce cuál será la próxima Prueba a realizar.
   - ❑ Se necesita obtener retroalimentación rápida de cierto producto o funcionalidad.
   - ❑ Se quiere investigar y aislar un defecto en particular.
   - ❑ Se quiere investigar el estado de un riesgo en particular.
   - ❑ Se quiere evaluar la necesidad de diseñar pruebas para un área en específico.
   - ❑ Otras, diga cuál:

     _____

     _____

12. Tiene definido su proyecto alguna Estrategia de Pruebas Exploratorias ó Testing Exploratorio.
    - ❑ Sí □ No □No sé

    b) Utiliza alguna (s) herramienta (s) para la ejecución de las Pruebas Exploratorias ó Testing Exploratorio.
       - ❑ Sí □ No □No sé

       En caso de ser su respuesta afirmativa, diga cuál (es):

Son de gran interés las respuestas que nos ha proporcionado. Muchas gracias.

Anexo 4. Análisis cuantitativo de la encuesta # 1

Planificación y seguimiento

Sí se planifica en el cronograma del proyecto las pruebas de software, sin embargo:

- no se tiene identificado el momento exacto que se entregarán los artefactos a Calisoft para su liberación.
- solamente el 54.81% de los encuestados ve “adecuada” la aplicación de las pruebas de software en su proyecto.
- no se cuenta con un cronograma específico, donde se identifiquen qué tipo de prueba hacer en cada momento.
- no se puede dar seguimiento al progreso del proyecto (cronogramas de pruebas, puntos de chequeo, etc.).
- falta de objetividad en la apreciación de la evolución de las pruebas, al no poseer los conocimientos y mecanismos necesarios para detectar la magnitud del trabajo ha desarrollar.
- los proyectos concluyen el desarrollo de sus artefactos, casi siempre, fuera de fecha, lo que reduce considerablemente el tiempo que se había planificado para la ejecución de las pruebas.

Calidad y productividad

- No se detectan con antelación los problemas en el software, afectando esto finalmente la calidad en los productos.
- Un 72 % de los proyectos productivos de la UCI realizan pruebas al finalizar el desarrollo de los componentes, subsistema o producto.
- Disminuye la capacidad del proyecto de reducir los costos al no pevenir los errores antes que se produzcan.
- Desconocimiento de los elementos y las características que integran el concepto de calidad de software.

- Solamente el 45.56 y 56.67 % de los encuestados seleccionaron bien las dos respuestas que identificaban cúales eran los elementos que conforman el concepto calidad de software.
- Desconocimiento de los tipos de pruebas que el equipo de proyecto le puede aplicar a sus componentes, subsistema o producto durante el desarrollo de los mismos.
- Un 58.15% respondió que se realizaban pruebas de validación y un 67.41% de liberación. Estas pruebas sí se le realizan a los artefactos generados por los poyectos, pero son guiadas y ejecutadas por Calisoft, no son pruebas internas del poyecto.
- El 53.33% respondió que le realizaban Revisiones Técnicas Formales (Pressman 2005), aunque éstas representan el filtro más efectivo desde el punto de vista de garantía de calidad y tienen como objetivo encontrar los errores durante el proceso, no constituyen un tipo de prueba en sí.
- Insuficiente aplicación de las pruebas de software en los proyectos, el 45.19% de los encuestados lo corrobora. Las principales causas por la que esto ocurre son:
    - ausencia de una adecuada estructura organizativa en el proyecto para este efecto, (opina el 11.48%).
    - desconocimiento del tema o técnicas a aplicar, (opina el 18.89%).
    - escaso personal capacitado y disponible para ello, (opina el 39.26%).
    - Pocos proyectos tienen definida alguna estrategia de PE ó Testing Exploratorio, (14.07% de los encuestados).
- Las PE se definen como el proceso simultáneo de exploración del producto (aprendizaje), diseño y ejecución de las mismas (Bach 2002). Solamente el 27.04% las aplica.
- Solamente el 4.81% de los proyectos encuestados expresan que utilizan alguna herramienta para la ejecución de las PE.
- Déficit de recursos humanos en cantidad y calidad vinculados a las pruebas en los proyectos.

Organizativas

- Mala concepción y ejecución de los procedimientos, procesos o metodologías que guían las pruebas durante el desarrollo de los proyectos.
- El 65.19% de los encuestados expresan que existe en sus poyectos un procedimiento, proceso o metodología que guíe las pruebas, y un 87.78% y 47.78% dicen que se le aplican pruebas de funcionalidad y usabilidad a sus artefactos respectivamente, sin embargo, solamente el 15.19% comenta que realiza pruebas de regresión (Pressman 2005a), éstas son de vital importancia durante el desarrollo del producto, pues consisten en volver a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse de que los cambios no han propagado efectos colaterales no deseados.
- Pobre formación y superación continua de los recursos humanos en temas de calidad.
- Aunque el 67.78% de los proyectos tiene un equipo dedicado al Aseguramiento de la Calidad (SQA) solamente el 36.30% expresó que las pruebas son “tan importantes como necesarias”. Esto evidencia que los directivos y jefe de proyecto o de equipo (61.48% de los encuestados) no le dan toda la importancia que necesita el tema de calidad en el desarrollo de sus productos.

Anexo 5. Análisis de los indicadores y sub-indicadores asociados a la variable eficiencia

Tabla 14: Análisis de los indicadores y sub-indicadores asociados a la variable eficiencia.

Indicadores	Sub-indicadores
Rendimiento	Optimización de recursos Esfuerzo dedicado a la actividad de prueba
Eficacia	Calidad de las pruebas
Tiempo	Tiempo de ejecución de las pruebas

Anexo 6. Laboratorios de pruebas de software en Latinoamérica y demás regiones del mundo

Tabla 15: Laboratorios de pruebas de software.

Laboratorios de Pruebas de Software	Institución	País
Test Center of Excellence (Hexaware Technologies)	Hexaware Technologies http://www.hexaware.com/	India
HP Software Virtual Lab (Hexaware Technologies 2011)	Hexaware Technologies and HP Software	México
ParqueSoft (ParqueSoft 2009)	Red de Parques Tecnológicos de Software, ParqueSoft Nation.http://www.parquesoft.com/	Colombia
Laboratorio de desarrollo y pruebas de software en el estado (Concyteg)	Concyteg. http://www.concyteg.gob.mx	México
Choucair Testing S.A.(Choucair Testing SA)	Choucair Testing http://www.choucairtesting.com	Colombia
e-Quallity (Equallity.com)	e-Quallity http://www.e-quallity.net	México
Laboratorio de Pruebas Funcionales. Centro de Ensayos de Software de Uruguay, (CES) (CES 1989)	Cámara Uruguaya de Tecnologías de la Información e Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República de Uruguay (UdelaR), a través de la Fundación Julio Ricaldoni. http://www.ces.com.uy	Uruguay
Laboratorio Nacional de Calidad de Software, INTECO (LNCS)	Instituto Nacional de Tecnologías de la Información http://www.inteco.es	España
Software Control & Testing,Sogeti (Sogeti 2009)	Grupo CapGemini http://www.es.sogeti.com/	España

Centro de Pruebas de Excelencia (Test Center of Excellence) (Hexaware Technologies).
Hexaware es un proveedor global de servicios de TI y de procesos de outsourcing. Se centra exclusivamente en maximizar la rentabilidad del cliente de la subcontratación y la deslocalización. Tiene una amplia experiencia en la gestión de grandes aplicaciones de TI en tiempo real, así como en la prestación de servicios de alto valor en torno a las aplicaciones empresariales estándar como SAP y PeopleSoft.

Hexaware le proporciona a las empresas un importante ahorro de costos. Nuestros centros de desarrollo se evalúan al SEI CMMI nivel 5, y son también la norma ISO 9001:2000 y certificado TickIT. Estos les ayuda a ofrecer continuamente un alto valor, los resultados de alta calidad a

nuestros clientes. Siempre se han entregado a las expectativas del cliente y han establecido relaciones duraderas con ellos, para ello cuenta con un Centro de Pruebas de Excelencia (TCOE).

Las mejores prácticas identificadas en el TCOE se propagarán a través de todos los proyectos asignados a las solicitudes del cliente. Hexaware y el cliente va a utilizar los modelos de pruebas construidos a partir de experiencias obtenidas en los últimos años, con vencimiento en TCOE. La aplicación de esta metodología permitirá a las entregas y menor costo del cliente.

Para ello establece que es imprescindible establecer una estrategia de pruebas, definir procesos, infraestructura, herramientas y técnicas, establecer la gobernabilidad, la métrica y el Service Level Agreement (SLA), el marco de la comunicación, el plan de gestión del conocimiento para atender las demandas de recursos y al modelo de prestación de configuración centralizada.

Principales beneficios del TCOE:

- Riesgo mitigado y se aceleró la madurez.
- Aprovechamiento de la innovación.
- SLA, servicios basados en las entregas.
- Multi-vendedor de la colaboración.
- Dominio centrado en la solución de pruebas personalizadas.
- Enfoque en el crecimiento del negocio del cliente.
- Socio estratégico.

HP Software Virtual Lab (Hexaware Technologies 2011)

Hexaware Technologies Limited., un proveedor líder global de TI, servicios de BPO y consultoría, ha anunció el 3 de marzo de 2011 que la Compañía ha puesto en marcha Hexaware HP Software Lab en su centro en México.

Este laboratorio de software mostrará el potencial de la suite de soluciones de HP Software que atienden a las necesidades del negocio en el rendimiento de validación, aplicación de seguridad, garantía de calidad y de negocios. Al aprovechar este laboratorio, Hexaware deberá demostrar sus herramientas de software propietario y los integradores de aplicaciones que la compañía ha desarrollado específicamente para ahorrar tiempo de implementación, los clientes maximizar el retorno de la inversión y facilitar el uso fácil de las soluciones líderes en la industria.

Este laboratorio de software servirá como puerta de entrada para demostrar la aplicación práctica de la solución de Hexaware de OneSource desarrollado para hacer frente a la completa aplicación de gestión del ciclo de vida con soluciones de HP Software. Las características más destacadas del laboratorio incluyen la demostración de la única prueba de aceleradores de Hexaware, las soluciones de firma electrónica, la conducta del cliente de prueba de conceptos y demostraciones en tiempo real a clientes a nivel mundial.

Hexaware ha optado por establecer este laboratorio de software cerca del centro de entrega global con sede en Saltillo, México. Al estar cerca de la costa su centro de entrega global ofrece una doble ventaja de hacer frente a los clientes con sede en EE.UU. y al mismo tiempo, abre la puerta al rápido crecimiento de América Latina de TI del mercado. A finales de 2010, Hexaware tiene 150 más empleados que operan en México.

ParqueSoft (ParqueSoft 2009)
ParqueSoft es uno de los principales proveedores de soluciones, productos y servicios en Tecnologías de la Información (TI) y relacionadas de América Latina.

Actualmente ParqueSoft en su Red de Parques Tecnológicos de Software, ParqueSoft Nation, alberga a más de 300 empresas especializadas en la industria del conocimiento, formando una comunidad de más de mil profesionales, desarrollando productos y servicios de conocimiento.

Más de 500 clientes satisfechos en todos los sectores de la economía, localizados en Estados Unidos, América Latina, Asia, Europa y África, confirman el potencial de innovación, investigación aplicada, utilización de tecnologías de punta, calidad en sus productos, servicios y procesos de gestión y soporte post venta que posee ParqueSoft.

Se han construido sólidas alianzas con las universidades nacionales e internacionales y con importantes centros de investigación, que hoy permiten tener en operación laboratorios de investigación alrededor de las Ciencias de la Computación y las Tecnologías de la Informática y las Comunicaciones. Estos laboratorios permiten a ParqueSoft obtener conocimiento en las Tecnologías Informáticas de punta y construir entornos más productivos y competitivos para el desarrollo de software y aplicación de conocimientos profundos de las ciencias de la computación en sus productos y servicios informáticos.

ParqueSoft cuenta con un modelo estratégico de apoyo al desarrollo de su comunidad a través de 5 estrategias y transversales, que permiten dar soporte de manera integral a cada uno de los emprendimientos y afinar su modelo de desarrollo y sus objetivos de construir una oportunidad de país entorno a la Ciencia y la Tecnología Informática. Una de las estrategias esenciales que se definen es la de calidad, cuyos objetivos son:

- Conformar equipo humano, selección y formación de un equipo humano experto en pruebas funcionales de software.
- Definición y consolidación de la metodología de pruebas funcionales de software adaptada a la cultura y a las necesidades de la industria en ParqueSoft.
- Crear un laboratorio, para ambientes de pruebas controladas.
- Herramienta, desarrollo e implementación de una herramienta de software de apoyo y complemento al trabajo de SQA, (Gómez).

ParqueSoft tiene como meta para el año 2012 desarrollar más de 1,000 empresas de Tecnología Informática y relacionadas, competitivas y productivas que exporten sus productos y servicios a los

mercados internacionales, generando más de 6,000 nuevas posiciones de trabajo permanentes en un nuevo sector innovador para Colombia.

Laboratorio de desarrollo y pruebas de software en el estado (Concyteg)
El laboratorio de desarrollo y pruebas de software en el estado fue creado por Concyteg en el 2007 como uno de sus proyectos estratégicos.

Su objetivo: Brindar servicios de desarrollo de software que cumplan con los estándares internacionales reconocidos, así como servicios de pruebas de software en calidad, compatibilidad, accesibilidad y funcionalidad. Lograr la participación en proyectos productivos en vinculación con la industria de TIC y mediante la innovación y la participación en proyectos multidisciplinarios que impacten a las diferentes cadenas productivas y generen conocimiento.

Mercado meta: Cualquier organización con requerimientos de software o que necesite realizar pruebas al software que desarrolla o adquiere.

Fábricas de software: Formación de nuevas fábricas y actualización de las ya existentes. Equipamiento condicionado mediante convenio a la participación en proyectos, replicación de cursos, etc.
Líneas estratégicas:

- Capacitación técnica.
- Capacitación en calidad y procesos.
- Desarrollo humano.
- Inglés con basamento matemático.

Choucair Testing S.A. (Choucair Testing SA)
Empresa colombiana con proyección internacional creada en el año 1999 dedicada exclusivamente a la prestación del servicio de pruebas de software (Software Testing) y convencida que la investigación y el recurso humano calificado, apoyados en una metodología certificada y efectiva, son los medios para prestar un servicio confiable tendiente a la disminución de los riesgos en operación, dentro del proceso de creación del software.

Durante sus diez años de existencia, ha trabajado con éxito más de 6.900 proyectos de pruebas, con excelentes resultados. La experiencia adquirida en los sectores financiero, telecomunicaciones y ERP, le han permitido a Choucair Testing, obtener ventajas como rapidez en la puesta en operación, conocimiento de los temas relacionados y experiencia en la resolución de problemas. Se especializa en el servicio de pruebas en los niveles denominadas de pruebas de sistema y de aceptación, en las diversas etapas del ciclo de desarrollo. Definen que un adecuado proceso de pruebas debe ser realizado por personal especializado e independiente que a través de un acompañamiento efectivo dentro del proceso, garantice el conocimiento y la objetividad. Su slogan esencial es “probamos lo que es importante para usted”.

Choucair divide sus servicios en dos grandes áreas:

- Operación y gestión de pruebas de software.
- Capacitación en cursos ISTQB.

Esta organización es la líder del HASTQB (Hispanic America Software Testing Qualification Board); y prestar el servicio de cursos de certificación para los niveles básicos, avanzados y expertos; diseñados y evaluados por ISTQB (International Software Testing Qualification Board), de la cual es miembro.

e-Quallity (Equallity.com)
Es un consorcio mexicano que se especializa en prueba de software. Utilizando como recurso central la prueba de software, e-Quallity Corp. es un socio de negocios de organizaciones que buscan desarrollar o adquirir productos de software de calidad, generados en el tiempo y con el presupuesto planeado y ven en ello una ventaja competitiva que les permite reducir riesgos y costos en el desarrollo de software. Para lograr su misión, e-Quallity Corp. desarrolla y adapta procesos, métodos, técnicas y herramientas.

Laboratorio de Pruebas Funcionales. Centro de Ensayos de Software de Uruguay, (CES) (CES 1989)
El CES, Centro de Ensayos de Software, es una organización especializada en proveer servicios de testing a empresas de tecnologías de la información y de otras industrias que la utilizan intensamente, permitiendo el incremento de su capacidad productiva mediante la mejora en la calidad, diversidad de plataformas e innovación de sus productos de software.

Se trata de un emprendimiento conjunto que llevan adelante la Cámara Uruguaya de Tecnologías de la Información (Cuti) y el Instituto de Computación (InCo) de la Facultad de Ingeniería de la Universidad de la República (UdelaR), a través de la Fundación Julio Ricaldoni.

El CES cuenta con un Laboratorio de pruebas para la evaluación de productos desde el punto de vista funcional, un Laboratorio de Ensayos de Plataformas para realizar pruebas de desempeño y resolver problemas de funcionamiento en arquitecturas de hardware y software complejas y un área administrativa-comercial donde se llevan a cabo las actividades de comercialización, administración y gestión.

Los laboratorios de testing están integrados por profesionales seleccionados entre los docentes del Instituto de Computación de la Facultad de Ingeniería (UdelaR), ingenieros y estudiantes avanzados de la carrera Ingeniería en Computación de dicha universidad.

Es un equipo especialmente capacitado, con vocación y entrenamiento para la resolución de problemas inéditos.

Laboratorio Nacional de Calidad de Software, INTECO (LNCS).

La misión del Laboratorio Nacional de Calidad del Software (LNCS) es promover estrategias de calidad en la industria TIC española, así como el desarrollo y asentamiento de proyectos empresariales en España. A tal fin se desarrollarán acciones encaminadas a:

- Promover la calidad y certificación de producto en el ámbito de las TIC mediante acciones de apoyo y estructuración del mercado (consorcios y asociaciones empresariales e institucionales).
- Aportar un valor añadido a los procesos de certificación de calidad de producto introduciendo el consenso sobre la certificación de productos en el mercado TIC.
- Reforzar la visibilidad de la industria TIC en los mercados internacionales.
- Incrementar la capacitación de los profesionales en las TIC mediante cursos, herramientas y servicios de apoyo.
- Desarrollar servicios públicos destinados a pymes y a usuarios finales de las TIC en materia de buenas prácticas de calidad.
- Establecer acciones permanentes de formación, sensibilización y prestación de servicios a empresas, facilitando el desarrollo de estrategias de calidad.

Uno de los objetivos del Laboratorio Nacional de Calidad del Software de INTECO es poder ofrecer servicios de difusión, sensibilización y formación en este tipo de áreas con la clara misión de:

- Promocionar los estándares y normativas, metodologías y herramientas orientadas a la calidad en el ámbito de las tecnologías de la información y la comunicación.
- Mejorar el capital humano de las empresas del sector TIC para afrontar con garantías el conjunto de procesos y actividades necesarios para alcanzar niveles de calidad óptimos.
- Identificar al laboratorio como referente en el ámbito nacional de calidad TIC.

Software Control & Testing, Sogeti (Sogeti 2009)

Es una compañía especialista en Servicios Tecnológicos y Soluciones de valor añadido alrededor de los sistemas de información y las infraestructuras que los soportan. Forma parte del Grupo CapGemini, uno de los principales proveedores de servicios de consultoría, tecnología y outsourcing del mundo. Forma parte de un líder europeo, que reúne hoy a 20.000 colaboradores, repartidos en 15 países.

La oferta del Grupo Sogeti se centra alrededor de 4 líneas de negocio:

- Servicios de aplicaciones (AS) - 41%.
- Servicios de infraestructura (IS) - 25%.
- Control & pruebas de software (SCT) - 18%.
- Consultoría de alta tecnología (HTC) - 16%.

Anexo 7. Tipos de pruebas según la ISO 9126

Tabla 16: Tipos de pruebas según la ISO 9126.

Característicade calidad	Tipos de Prueba
Funcionalidad	Funcionalidad: Consisten en la revisión de los requisitos aceptados por el cliente contra las funcionalidades presentes en la aplicación. Seguridad: Asegurar que los datos o el sistema solamente es accedido por los actores definidos según niveles de acceso. Volumen: Enfocada en verificar las habilidades de los programas para manejar grandes cantidades de datos, tanto como entrada, salida o residente en la Base de Datos.
Confiabilidad	Recuperación y tolerancia a fallas: Verificar que los procesos de recuperación (manual o automática) restauran apropiadamente la base de datos, aplicaciones y sistemas y los llevan a un estado conocido o deseado. Los siguientes tipos de condiciones deben incluirse en la prueba: interrupción de electricidad en el cliente, interrupción de electricidad en el servidor, interrupción en la comunicación hacia el servidor (caídas de red), interrupción en la comunicación con los controladores de disco, entre otros. Benchmark (Comparativa): Es un tipo de prueba que compara el rendimiento de un elemento nuevo o desconocido a uno de carga de trabajo de referencia conocido.
Usabilidad	Usabilidad: Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento. Estructura: Enfocada a la valoración a la adherencia a su diseño y formación. Este tipo de prueba se le ejecutan a las aplicaciones Web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.
Eficiencia	Contención: Enfocada a la validación de las habilidades del elemento a probar para manejar aceptablemente la demanda de múltiples actores sobre un mismo recurso (registro de recursos, memoria, etc.). Carga: Usada para validar y valorarla aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece

	constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales. Stress: Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible). Rendimiento: Enfocadas a monitorear el tiempo en flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccionar los cuellos de botellas y los procesos ineficientes.
Mantenibilidad	No se han definido tipos de prueba realizar durante la liberación, pues esta característica tiene condiciones particulares. Se está estudiando a fondo de qué forma se puede comprobar si las aplicaciones informáticas que se prueben, cumplen con los elementos esenciales que lo puedan hacer mantenible en el tiempo.
Portabilidad	Configuración: Enfocada a asegurar que funciona en diferentes configuraciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema. Instalación: Enfocada a asegurar la instalación en diferentes configuraciones de hardware y software bajo diferentes condiciones (insuficiente espacio en disco, etc.).

Anexo 8: Herramientas automatizadas para las pruebas

Más información se puede obtener en: (Rojas 2007).

Tabla 17: Herramientas automatizadas de apoyo a las pruebas.

Herramienta	Descripción
Ejecución de las pruebas	
JMeter	Herramienta de carga y rendimiento. Organización: The Apache Jakarta Project. Apache Jmeter es una aplicación de escritorio 100% java, diseñada para realizar pruebas de carga funcional y medición de rendimiento. Esta fue diseñada originalmente para probar aplicaciones Web, se ha ampliado desde entonces a otras funciones de prueba. (Apache Software Foundation 1999-2010) (http://jakarta.apache.org/jmeter/)
DBMonster	Utilizada para la carga masiva de datos. Pruebas de volúmen. (Geeknet 2011) (http://dbmonster.kernelpanic.pl/).
Selenium	Es una herramienta de Software Libre para pruebas de aplicaciones Web. Las pruebas se ejecutan directamente en un navegador y

	facilitan las pruebas de compatibilidad en navegadores, también como pruebas funcionales de aceptación de aplicaciones Web. Posee un ambiente de desarrollo llamado Selenium IDE, este facilita el registro de pruebas de aceptación y su depuración. (Badle 2010) (http://seleniumhq.org/).
Meld	Es una herramienta para comparar ficheros y buscar diferencias entre dos versiones distintas del mismo archivo. (Wasesores 2010-2011)
Virtualización del LIPS	
VMWare	Es una potente herramienta de virtualización probada en entornos de la universidad. Es multiplataforma. Permite la ejecución concurrente de máquinas virtuales. (VMware 2011) (http://www.vmware.com).
vSphere Client	Capacidad para configurar máquinas virtuales hasta 4 CPUs virtuales cada una. Soporte para hardware de servidores de 64 bits hasta 256 GB de memoria RAM física. Incluye la seguridad integrada de red y servidor de seguridad de protección para las máquinas virtuales. (VMware 2011a).
Control y seguimiento de las No Conformidades	
Redmine	Es una herramienta Web de gestión de proyectos de aplicaciones web. Es de código abierto y liberado bajo los términos de la GNU General Public License v.2. Soporta múltiples proyectos. Permite el seguimiento de defectos. (Lang 2006-2011) (http://www.redmine.org/).

Tabla 18: Descripción de las herramientas automatizadas para las pruebas.

Herramienta	Descripción	Se ejecutan en el LIPS (sí/no)
Cactus	Framework para realizar pruebas de unidad y de integración del lado del servidor. Está formado por varios componentes: Cactus Framework, Cactus Integration Modules y Cactus Samples.	No
jMeter	Herramienta de carga y rendimiento. Organización: The Apache Jakarta Project. Apache Jmeter es una aplicación de escritorio 100% java, diseñada para realizar pruebas de carga funcional y medición de rendimiento. Esta fue diseñada originalmente para probar aplicaciones Web, se ha ampliado desde entonces a otras funciones de prueba. (Apache Software Foundation 1999-2010) (http://jakarta.apache.org/jmeter/)	Sí
Load Runner Product Family	Herramientas de carga y rendimiento. Sistema multiusuario y herramienta para pruebas de servidor. Organización: Mercury Interactive. Herramienta para pruebas de Sistemas cliente/servidor. Permite realizar pruebas de rendimiento, pruebas de carga y afinación de aplicaciones para múltiples usuarios. (http://www.merc-int.com/)	No
e - Load	Herramientas de carga y rendimiento. Herramienta de pruebas para carga, stress y escalabilidad para aplicaciones Web. Organización: Empirix. Esta herramienta es un componente de la suite e-Test. Puede ser utilizada por los desarrolladores	No

	tempranamente en el proceso del desarrollo para validar la escalabilidad de la arquitectura en conjunto. Puede también ser utilizada para probar y poner a punto la aplicación completa bajo carga antes del despliegue. (http://www.empirix.com/)	
JUnit	Pruebas de unidad. Marco de trabajo para pruebas de regresión usado por los desarrolladores para realizar pruebas de unidad en java. Organización: JUnit.org. JUnit es un marco de trabajo de pruebas de regresión escrito por Erich Gamma y Kent Beck. Es utilizado por los desarrolladores para implementar pruebas de unidad en java. (http://www.junit.org/)	No
JWebUnit	Framework que facilita la creación de pruebas de aceptación para aplicaciones web. Incluye navegación a través de enlaces, entrada por formularios y envío de información a través de los mismos, validación de los contenidos de las tablas, y otras características relativas a las aplicaciones web.	No
TestNG	Framework de pruebas diseñado para simplificar el alto rango de pruebas necesarias. Brinda la posibilidad de realizar pruebas unitarias y pruebas de integración.	No
vTest	Herramienta automática para pruebas funcionales y de regresión para aplicaciones web. Incorpora capacidades de registro, verificación, reproducción y reporte. Soporta Internet Explorer y Mozilla Firefox. Soporta las tecnologías HTTP, HTTPS/SSL, Javascript, DHTML, ActiveX, Java y Flash, así como los entornos de aplicaciones web ASP, ASP.NET, Java Servlets y JSP, PHP y CGI. Sin ninguna programación, todos los objetos son capturados y registrados automáticamente como un programa gráfico, que muestra los pasos en su programa en forma de un árbol basado en iconos. Usa un algoritmo de identificación de objeto sofisticado para identificar los objetos en su aplicación.	No
DBMonster	Utilizada para la carga masiva de datos. Pruebas de volúmen. (Geeknet 2011) (http://dbmonster.kernelpanic.pl/).	Sí
Selenium	Es una herramienta de Software Libre para pruebas de aplicaciones Web. Las pruebas se ejecutan directamente en un navegador y facilitan las pruebas de compatibilidad en navegadores, también como pruebas funcionales de aceptación de aplicaciones Web. Posee un ambiente de desarrollo llamado Selenium IDE, este facilita el registro de pruebas de aceptación y su depuración. (Badle 2010) (http://seleniumhq.org/).	Sí
Meld	Es una herramienta para comparar ficheros y buscar diferencias entre dos versiones distintas del mismo archivo. (Wasesores 2010-2011)	Sí
Virtualización del LIPS		
VMWare	Es una potente herramienta de virtualización probada en entornos de la Universidad. Es multiplataforma. Permite la ejecución concurrente de máquinas virtuales. (VMware 2011) (http://www.vmware.com).	Sí
vSphere Client	Capacidad para configurar máquinas virtuales hasta 4 CPUs virtuales cada una. Soporte para hardware de servidores de 64	Sí

	bits hasta 256 GB de memoria RAM física. Incluye la seguridad integrada de red y servidor de seguridad de protección para las máquinas virtuales. (VMware 2011a).	
Control y seguimiento de las No Conformidades (Rojas 2007)		
Bugzilla	Seguimiento de defectos. Base de datos basada en Web para defectos. Organización: Mozilla. Es una herramienta de seguimiento de defectos que permite a los individuos o grupos de desarrollo no perder de vista los defectos encontrados en su producto. (http://www.bugzilla.org/).	No
Mantis	Es una herramienta Web de seguimiento de defectos. Fue realizado con lenguaje PHP y trabaja con MySql, MS SQL, PosgreSQL y un servidor Web. Fácil de instalar, creación de reportes, notificaciones a e-mail, entre otros. Organización: Mantis project. (http://mantisbt.sourceforge.net/).	No
Redmine	Es una herramienta Web de gestión de proyectos de aplicaciones web. Es de código abierto y liberado bajo los términos de la GNU General Public License v.2. Soporta múltiples proyectos. Permite el seguimiento de defectos. (Lang 2006-2011) (http://www.redmine.org/).	Sí
Rational Clear Quest	Seguimiento de defectos. Gestión de cambios. Organización: Software Racional de IBM. Herramienta de gestión de cambios y defectos que ofrece funciones de automatización de procesos, creación de informes, gestión de pruebas y seguimiento de cambios y defectos durante todo el proyecto. (http://www-306.ibm.com/software/info/ecatalog/es_ ES/products/J162883X27728 Y69.html).	No
DevTrack	Es una herramienta de seguimiento de defectos y del proyecto, diseñada específicamente para los equipos de desarrollo de software. Organización: TechExcel,inc. Seguimiento y manejo comprensivo de todos los defectos del producto, peticiones de cambio y otras fases del desarrollo. DevTrack facilita trabajo en equipo, entre los diversos usuarios e incluso clientes. También proporciona la automatización de gran alcance de tareas relacionadas y del proceso. (http://www.techexcel.com/).	No
Census Bug Tracking and Defect Tracking	Es una herramienta de seguimiento de defectos y fallos basada en Web altamente escalable que puede también seguir peticiones de cambio, llamadas de soporte, casos de prueba y mucho más. Las características incluyen capacidades completas de manejo de requisitos, la integración visual de SourceSafe, notificaciones automáticas a e-mail, niveles de seguridad para usuario/grupo, reglas basadas en el workflow, cliente Web para diferentes grupos de usuarios, creación de reportes, archivos adjuntos y seguimiento de la historia de cambio. Organización: MetaQuest Software inc.	No
BugRat	BugRat es un software en java, libre, que proporciona un sofisticado y flexible sistema de reporte y seguimiento de defectos. Proporciona las siguientes característica: -Utiliza base de datos relacionales para almacenamiento. -Utiliza JDBC para acceso a base de datos. -Permite reporte de defectos vía Web. -Permite reporte de defectos vía e-mail.	No

	-Permite navegación y búsqueda vía Web. -Incluye clientes java para manejo de la base de datos. Organización: Giant Java Tree. (http://www.gjt.org/pkg/bugrat/).	

Anexo 9. Técnicas de prueba. Descripción de las mismas

Tabla 19: Técnicas de prueba.

Agrupación	Técnicas
Técnicas de Caja Negra	Partición de equivalencia Análisis del valor límite Tablas de decisión Máquinas de estado finito Grafo causa efecto Prueba de casos de uso Prueba de dominios
Técnicas de Caja Blanca	Basadas en el flujo de control Basadas en el flujo de los datos Mutantes
Técnicas según quién hace la prueba	Pruebas de aceptación Pruebas alfa y beta Pruebas de usuario Pruebas en pares
Técnicas basadas en la experiencia	Prueba Ad hoc Conjetura de errores **Pruebas Exploratorias o Testing Exploratorio**

Técnicas de Caja negra

Partición de equivalencia

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

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

La primera consideración se utiliza para desarrollar un sistema mínimo de casos de prueba que cubren estas condiciones. La segunda consideración para desarrollar un sistema de condiciones "interesantes" que se probarán. El diseño de casos de prueba siguiendo partición de equivalencia se divide en dos pasos (Myers G. 2004):

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

Análisis del valor límite

La experiencia demuestra que los casos de prueba que exploran condiciones límites tienen una rentabilidad mayor. Las condiciones límite son situaciones en los bordes, por arriba y por debajo de las clases de equivalencia para los valores de la entrada y de la salida, (Myers G. 2004).

Tablas de decisión

Las tablas de decisión representan relaciones lógicas entre las condiciones (entradas) y las acciones (salidas). Los casos de prueba son derivados sistemáticamente considerando cada combinación posible de condiciones y de acciones, (SWEBOK 2004).

Son una buena manera de capturar los requerimientos del sistema que contienen condiciones lógicas. Pueden ser utilizadas para registrar reglas de negocio complejas de un sistema. Se analiza la especificación, las condiciones y las acciones del sistema se identifican. Las condiciones y las acciones de la entrada se indican de manera tal que puedan ser verdaderas o falsas (booleano). Cada columna de la tabla corresponde a una regla de negocio que define una combinación única de las condiciones que dan lugar a la ejecución de las acciones asociadas a esa regla. La idea es tener por lo menos una prueba por columna, que implica cubrir todas las combinaciones para accionar condiciones.

La fortaleza de la tabla de decisión es crear combinaciones de condiciones que pueden no ejercitarse de otra manera. Puede ser aplicada cuando la acción del software depende de varias decisiones lógicas, (ISTQB. 2005).

Máquinas de estado finito

Modelando un programa como máquina de estado finito, las pruebas se pueden seleccionar para cubrir estados y sus transiciones, (SWEBOK 2004).

Un sistema puede tener distintas respuestas dependiendo de las condiciones actuales o de la historia anterior (su estado). En ese caso, ese aspecto del sistema se puede mostrar como máquinas de estado.

Esta forma de modelar permite ver el software en términos de sus estados, las transiciones entre los estados, las entradas o los acontecimientos que disparan el cambio de estado (las transiciones) y las acciones que pueden resultar de esas transiciones. Los estados del sistema son identificables y finitos en número. Las pruebas se diseñan para cubrir una secuencia típica de estados, para cubrir cada estado, para ejercitar cada transición, para ejercitar secuencias específicas de transiciones o para probar transiciones inválidas. La técnica es conveniente para modelar un objeto del negocio que tiene estados específicos o flujos de diálogo de pantalla, (ISTQB. 2005).

Grafo causa efecto

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

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

La técnica de grafo causa-efecto no explora adecuadamente condiciones de límite. El aspecto más difícil de la técnica es la conversión del grafo en la tabla de decisión. Este proceso es algorítmico y se puede automatizar, (Myers G. 2004).

Prueba de casos de uso

Un caso de uso es una secuencia de acciones que un sistema realiza con el fin de lograr un resultado de valor para un actor particular. Un actor es alguien o algo fuera del sistema que interactúa con el sistema, (Jacobson 1999). Cuenta con precondiciones, que deben cumplirse para que el mismo se realice con éxito y pos-condiciones, que son los resultados observables y el estado final del sistema después de que se haya terminado el caso de uso. Un caso de uso tiene un escenario o flujo principal y varios flujos alternativos.

Las pruebas se pueden especificar a partir de casos de uso o escenarios del negocio. Los casos de uso describen los escenarios a través de un sistema basado en su uso probable real, por lo que los casos de prueba derivados a partir de ellos son muy útiles para encontrar defectos en los escenarios durante el uso real del sistema. Los casos de uso son muy útiles para diseñar pruebas de aceptación con la participación del cliente o del usuario y ayudan a descubrir defectos en la integración causados por la interacción de diversos componentes, que la prueba individual del componente no considera, (ISTQB. 2005).

Prueba de dominios

Un dominio es un conjunto que incluye todos los posibles valores de una variable para una función. En la prueba de dominio se identifican las variables y las funciones. Las variables pueden ser de entrada o de salida. Para cada una, se toman pocos valores representativos de los posibles de la clase de equivalencia (típicamente casos bordes) para cada clase. La hipótesis del método es que si se prueba con unos pocos elementos pero bien representativos de la clase, se encontrarán la mayoría de los defectos que se encontrarían si se prueba con cada miembro de la clase. El elemento primario de interés es la variable más que la función, la prueba de dominio debe analizar una variable y basado en ese análisis, ejecutar pruebas que involucren esta variable en cada función como entrada o salida, (Kaner C. 2001).

Técnicas de Caja Blanca

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

Basadas en el flujo de control

Los criterios basados en el flujo de control o la estructura del programa, cubren todas las sentencias o bloques de sentencias en un programa, o combinaciones especificadas de ellas. Se han propuesto varios criterios de cobertura. El más fuerte de los criterios es el de flujo de control, que ejecuta todas las trayectorias del flujo del control de la entrada a la salida. Puesto que la prueba de la trayectoria no es generalmente factible debido a los bucles, otros criterios menos rigurosos son utilizados en la práctica, por ejemplo de cobertura de sentencia, de condiciones y de decisión. La adecuación de tales pruebas se mide en porcentajes; por ejemplo, se dice haber alcanzado una cobertura de sentencia del 100% cuando todos las sentencias han sido ejecutadas por lo menos una vez por las pruebas, (SWEBOK 2004).

Basadas en el flujo de los datos

La prueba en el flujo de datos usa la información sobre cómo se definen, se utilizan y se destruyen las variables del programa. El criterio más fuerte, all definition-use paths, requiere que para cada variable, cada segmento de la trayectoria del flujo del control desde una definición de esa variable hasta su uso sea ejecutado. Para reducir el número de las trayectorias requeridas se utilizan estrategias más débiles como all-definitions y all-uses, (SWEBOK 2004).

Mutantes

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

mutantes que sobreviven. Para que la técnica sea eficaz, se deben derivar automáticamente de una manera sistemática una gran cantidad de mutantes, (SWEBOK 2004).

Técnicas según quién hace la prueba

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

Prueba de aceptación

La prueba de aceptación es el proceso de comparar el programa contra sus requerimientos iniciales y las necesidades reales de los usuarios. Es realizado generalmente por el cliente o el usuario final. En el caso de un programa por contrato, se realiza la prueba de aceptación comparando la operación del programa contra el contrato original, (Myers G. 2004).

Generalmente se realiza seleccionando un subconjunto de los casos de prueba del sistema que demuestra formalmente las funcionalidades claves para su aprobación final. Es realizado comúnmente luego que el sistema es instalado en el ambiente del usuario, (Daich 1994).

Pruebas alfa y beta

Antes de que el software se libere, se distribuye a un pequeño grupo representativo de los usuarios potenciales para el uso interno (alfa) o externo (beta). Estos usuarios reportan problemas con el producto. El uso de las pruebas alfa y beta es a menudo descontrolado y no siempre es citado en el plan de prueba, (SWEBOK 2004)

Prueba de usuario

Es la prueba realizada por el tipo de persona que usará el producto. Puede ser realizado en cualquier momento durante el desarrollo, en el lugar del cliente o en el de desarrollo, en ejercicios completamente dirigidos o a la discreción del usuario, (Kaner C. 2001).

Prueba en pares

Consiste en que dos probadores trabajan juntos para encontrar errores. Comparten una computadora e intercambian el control de la misma mientras prueban, (Kaner C. 2001).

Técnicas basadas en la intuición o en la experiencia

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

Prueba Ad hoc

Quizás la técnica más practicada sigue siendo la prueba ad hoc: las pruebas son derivadas confiando en la habilidad, intuición y experiencia con programas similares. La prueba ad hoc puede ser útil para identificar pruebas que no son fácilmente encontradas por técnicas más formales, (SWEBOK 2004).

Conjetura de errores

Se ha observado a menudo que ciertas personas sin usar ninguna metodología particular, parecen tener una destreza para encontrar los errores. Dado un programa particular, conjeturan, por la intuición y la experiencia, ciertos tipos probables de errores y después escriben casos de prueba para exponer esos errores. La idea básica es enumerar una lista de errores y después escribir los casos de prueba basados en la lista. Otra idea es identificar los casos de prueba asociados a asunciones que el programador pudo haber hecho cuando leía la especificación (es decir, las cosas que fueron omitidas de la especificación, por accidente o porque eran obvias), (Myers G. 2004).

Los casos de prueba son diseñados pensando las faltas más probables de un programa dado. Una buena fuente de información es la historia de las faltas descubiertas en proyectos anteriores, así como la experiencia de quien prueba, (SWEBOK 2004).

Anexo 10. Criterios de criticidad establecidos por el DPSW

Los Criterios de Criticidad establecen los parámetros para declarar un producto del desarrollo de software en estado crítico de terminación.

Prueba Detenida (PD): Se detiene la prueba y se reinicia en la misma actividad que se detuvo. Como máximo una prueba puede estar detenida 1 semana, este plazo se decide según los motivos que hicieron que se detuviera y debe cumplirse por parte del equipo de desarrollo. En caso de que no se cumpla el tiempo, se declara Prueba Abortada. Criterios de criticidad para pruebas detenidas:

- No se presentan todos los artefactos de apoyo requeridos para realizar la prueba (RI).
- No está lista la última versión del artefacto que debe liberarse (RI), (PEI), (IP).
- Se mantienen NC detectadas en iteraciones anteriores (PEI), (IP), (PEP).

Prueba Abortada (PA): Se detiene la prueba y para reiniciarla debe hacerse una nueva solicitud, comenzando el proceso desde el inicio. Como mínimo se necesitan 3 días para recomenzar la prueba. Criterios de criticidad para pruebas abortadas:

- No están presentes todos los elementos componentes del sistema, producto o entregable (hardware, software, documentación y artefactos de apoyo), (RI), (PEI), (IP), (PEP).
- No se corresponden totalmente los requisitos funcionales documentados con los implementados (PEI), (IP), (PEP).
- Supera el artefacto la cantidad de NC Significativa por unidad de revisión, (*las unidades de revisión pueden ser funcionalidades, elementos esenciales de la metodología, cantidad de páginas definidas, etc.),* (PEI), (PEP).
- Supera el producto la taza de 1 NC significativa por caso de prueba o página, (IP).
- Excede el producto las 3 iteraciones establecidas en la planeación inicial, (IP).
- Se incumplen las reglas para la documentación establecidas por la entidad y/o el proyecto, (PEI), (IP), (PEP).
- Existe incoherencia entre los artefactos que tienen relación o dependencia entre sí, (PEI), (IP),

(PEP).

- Existe incoherencia entre lo documentado y lo implementado, (PEI), (IP), (PEP).
- Se incumple con lo pactado en el plan de pruebas, (PEI) (IP), (PEP).
- Se excede el tiempo total de la prueba según lo planificado en el cronograma pactado en el plan de pruebas, (PEI) (IP), (PEP).

Anexo 11. Plantilla de solicitud de las pruebas de liberación

Fecha de la solicitud:
Centro:
Nombre del proyecto:
Nombre del producto:
Versión:
Descripción del proyecto:
Artefactos a liberar:

Artefacto	Tipo de artefacto	Versión	Cant. Páginas	Cant. Requisitos o CU Complejidad Alta	Cant. Requisitos o CU Complejidad Media	Cant. Requisitos o CU Complejidad Baja

Datos para el entorno de pruebas:

	Servidor Aplicación	Servidor Base de Datos	Cliente
Capacidad disco duro (en GB)			
Memoria RAM			
Sistema Operativo			

Navegador Web:
Gestor de BD:
Otros requisitos de HW y SW:
Tipos de prueba:

Anexo 12. Principio, diagrama y gráfica de Pareto

El nombre de Pareto fue dado por el Dr. Juran en honor del economista italiano Vilfredo Pareto (1848-1923), quien realizó un estudio sobre la distribución de la riqueza, en el cual descubrió que la minoría de la población poseía la mayor parte de la riqueza y la mayoría de la población poseía la menor parte de la riqueza, (Rovira).

Pareto es una herramienta de análisis de datos ampliamente utilizada y es por lo tanto útil en la determinación de la causa principal durante un esfuerzo de resolución de problemas. Esto permite ver cuáles son los problemas más grandes, permitiéndoles a los grupos establecer prioridades. En estos casos típicos, los pocos (tipos de no conformidades) son los responsables por la mayor parte del impacto negativo sobre la calidad del artefacto. Si el equipo de proyecto enfoca su atención en estos

pocos vitales, puede obtener la mayor ganancia potencial de sus esfuerzos por mejorar la calidad de su producto.

Para entender este principio se muestra a continuación la Figura 13.

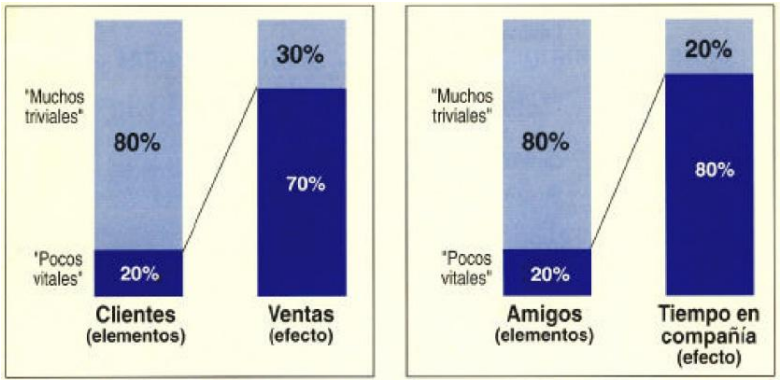


Figura 13: Principio de Pareto.

El Diagrama de Pareto constituye un sencillo y gráfico método de análisis que permite discriminar entre las causas más importantes de un problema (los pocos vitales) y las que lo son menos (los muchos triviales). En realidad este diagrama prioriza los problemas que se tienen en la organización de mayor a menor.

Tablas y diagrama (gráfica) de Pareto

Las tablas y diagramas de Pareto son herramientas de representación utilizadas para visualizar el análisis de Pareto. El diagrama de Pareto es la representación gráfica de la tabla de Pareto correspondiente. Las etapas que se deben completar para construir un diagrama de Pareto son:

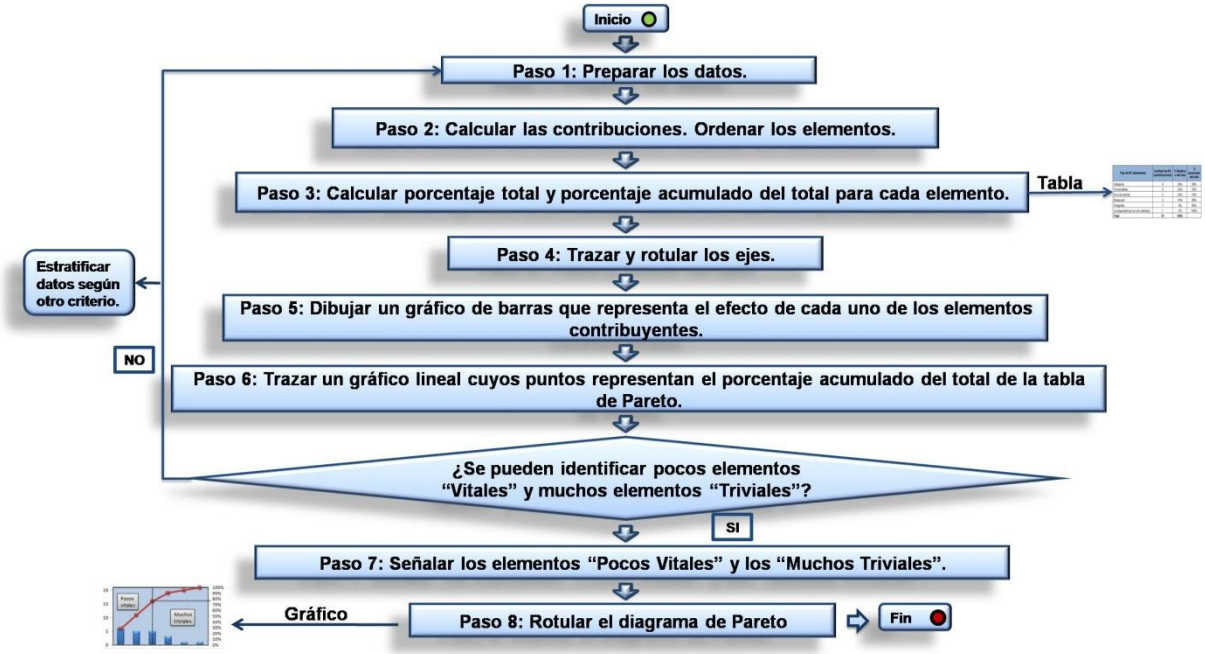


Figura 14: Etapas para la construcción del diagrama de Pareto.

Se presenta el primer ejemplo de la utilización del diagrama de Pareto, con él se pretende determinar cuáles son las principales no conformidades que inciden sobre el artefacto probado. El artefacto es una aplicación web que cuenta con un total de 9 DCP al cual se le realizó pruebas exploratorias y la evaluación final del artefacto fue abortada.

A continuación se muestran los resultados del cálculo de las contribuciones, cómo se calcula el porcentaje total y porcentaje acumulado del total para cada elemento (la suma de todos los porcentajes debe ser igual al 100%).

El porcentaje de la contribución de cada elemento (% total) se calcula:
% = (magnitud de la contribución / magnitud del efecto total) x 100

El porcentaje acumulado del total para cada elemento de la lista ordenada se calcula:

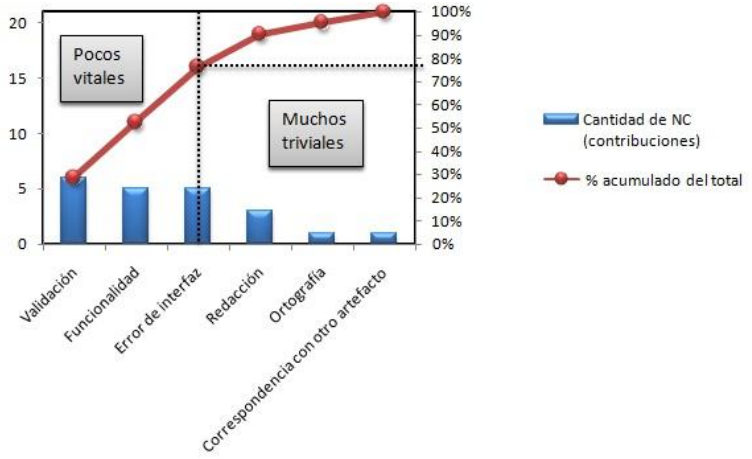
- Por suma de contribuciones de cada uno de los elementos anteriores en la tabla, más el elemento en cuestión como magnitud de la contribución y aplicando la fórmula anterior.
- Por suma de porcentajes de contribución de cada uno de los elementos anteriores más el porcentaje del elemento en cuestión. En este caso habrá que tener en cuenta el que estos porcentajes, en general, han sido redondeados.

En la tabla a continuación se muestran los datos:

Tabla 20: Cálculo porcentaje total y porcentaje acumulado del total para cada elemento, ejemplo 1.

Tipo de NC (elementos)	Cantidad de NC (contribuciones)	% del total	% acumulado del total
Validación	6	29%	29%
Funcionalidad	5	24%	52%
Error de interfaz	5	24%	76%
Redacción	3	14%	90%
Ortografía	1	5%	95%
Correspondencia con otro artefacto	1	5%	100%
Total	**21**	**100%**	

En la Gráfica 7 que se muestra a continuación se presenta la gráfica de Pareto con la tendencia de las NC del artefacto que se probó. Esta gráfica contiene 2 gráficos solapados, barras y línea. Las barras representa la frecuencia reincidente de las no conformidades que inciden sobre el artefacto, encontradas durante la ejecución de las pruebas, las mismas se agrupan de forma descendiente. La gráfica de línea representa el porcentaje acumulado del total de ocurrencias de las no conformidades y se gráfica de forma decreciente como una función cóncava.



Gráfica 7: Gráfica de Pareto para el ejemplo 1.

El especialista de prueba le proporciona al equipo de desarrollo la interpretación y análisis de los datos resultantes obtenidos de la implementación del diagrama de Pareto con su gráfica respectiva. El proyecto puede utilizar estos datos para varios propósitos: analizar las causas que dieron origen a las no conformidades, estudiar los resultados y planear una mejora continua.

Lo importante no es el porcentaje más o menos desproporcionado que se obtenga, sino que las barras con mayor porcentaje, las que aparecen a la izquierda, son las que representan los problemas más acuciantes y cuanto más a la izquierda el problema es más importante y de mayor urgencia su resolución.

La interpretación de esta gráfica de Pareto es: existen 21 errores relacionados con las no conformidades emitidas durante las PEI. Pero de estos 16 que son del tipo validación, funcionalidad y error de interfaz (pocos vitales), corresponden al 75% del total de no conformidades. Por lo tanto el equipo de desarrollo debe enfocarse mayormente en la resolución de éstas 16 no conformidades de categoría pocos vitales, puesto que representan la mayor ganancia potencial para mejorar la calidad del sistema, es decir, dirigiendo sus esfuerzos a estas no conformidades solucionan el 80% de los errores de su producto.

El segundo ejemplo que se presenta corresponde a la tendencia de las no conformidades en artefactos del mismo tipo, lo cual permitirá al laboratorio de prueba tener una visión de las posibles no conformidades que se encontrará en un artefacto x. Esto además permite antes de comenzar una iteración de prueba, o las PEI, enfocar el esfuerzo de las pruebas a las funcionalidades propensas a fallar, según los resultados que emita la gráfica de Pareto.

Para este ejemplo se seleccionó 11 proyectos todos artefactos de aplicación web, que fueron probados en el LIPS. A continuación se muestra la Tabla 21 con las no conformidades que se emitieron durante las pruebas.

Tabla 21: Cantidad de No conformidades por tipo de error.

	Cantidad de No conformidades por tipo de error														
Proyecto	**Cant. de DCP, requisitos, funcionalidades, CU**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
A	31		1			2				1					
B	10	1											4		
C	14	3	1					1	2		7				
D	19					3				3					
E	36	4	1	1		2		6	1	11				2	13
F	18	6	1	4	1		4		4	4	1	2		2	
G	4				1		1			1		2			
H	21	20	20		8	11				80				2	
I	1			4		2				5				2	
J	29					47		4		19		5		16	
K	63	18	1			6	13	6		4	1	2		17	
Totales	**246**	**52**	**25**	**9**	**10**	**73**	**18**	**16**	**7**	**128**	**9**	**11**	**4**	**41**	**13**

Leyenda:

1. Ortografía
2. Formato
3. Redacción
4. Error técnico
5. Validación
6. Errores de Interfaz
7. Correspondencia con otra documentación
8. Errores de Idioma
9. Funcionalidad
10. Excepciones
11. Otros Errores
12. Seguridad
13. Recomendaciones
14. Diseño Casos de prueba

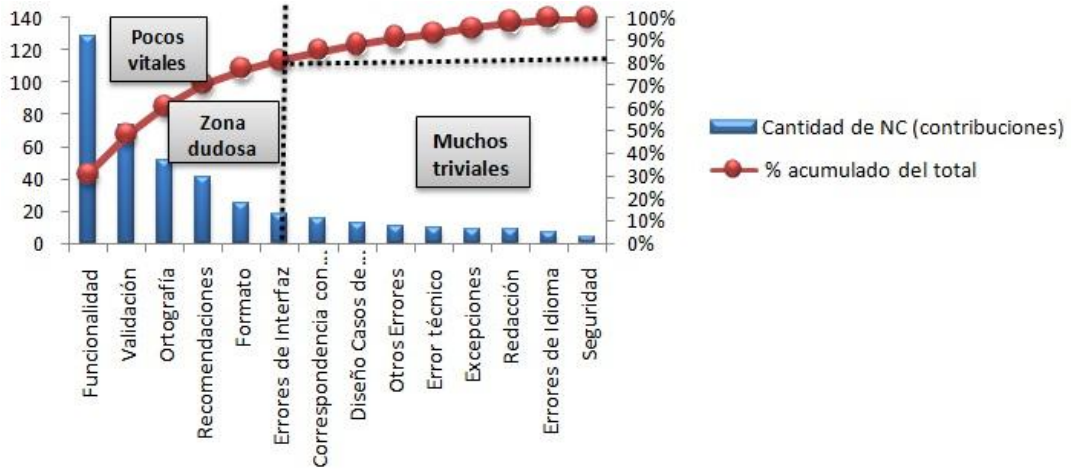
Una vez tabulados los datos de las no conformidades por tipo de error, se procede al cálculo de las contribuciones, el porcentaje total y porcentaje acumulado del total para cada elemento (la suma de

todos los porcentajes debe ser igual al 100%), paso 2 y 3 de la elaboración del Diagrama de Pareto (ver Figura 14). A continuación se muestran los resultados:

Tabla 22: Cálculo porcentaje total y porcentaje acumulado del total para cada elemento, ejemplo 2.

Tipo de NC (elementos)	Cantidad de NC (contribuciones)	% del total	% acumulado del total
Funcionalidad	128	31%	31%
Validación	73	18%	48%
Ortografía	52	13%	61%
Recomendaciones	41	10%	71%
Formato	25	6%	77%
Errores de Interfaz	18	4%	81%
Correspondencia con otra documentación	16	4%	85%
Diseño Casos de prueba	13	3%	88%
Otros Errores	11	3%	91%
Error técnico	10	2%	93%
Excepciones	9	2%	95%
Redacción	9	2%	97%
Errores de Idioma	7	2%	99%
Seguridad	4	1%	**100%**
Total	**416**	**100%**	

En el Gráfica 8 que se muestra a continuación se presenta la gráfica de Pareto con la tendencia de las NC de los artefactos aplicación web de diferentes proyectos que fueron probados por especialistas del LIPS.



Gráfica 8: Gráfica de Pareto para el ejemplo 2.

La interpretación de esta gráfica de Pareto es: existen 416 errores relacionados con las no conformidades emitidas durante las pruebas de estos artefactos. Pero de estos, 337 que son del tipo

funcionalidad, validación, ortografía, formato y error de interfaz (pocos vitales), corresponden al 80% del total de no conformidades. Los resultados arrojados le permiten al equipo de prueba realizar diferentes análisis:

1. saber cuáles son las funcionalidades sobre las que debe dirigir mayormente las pruebas, en las que se puede encontrar las no conformidades que están dentro de la categoría pocos vitales, puesto que representan el mayor número de errores en los artefactos aplicación web, es decir, dirigiendo el esfuerzo de las pruebas a encontrar estas no conformidades, se localiza el 80% de los errores del producto.
2. proporciona un histórico para cualquier análisis que se le pida al DPSW sobre las tendencias de las no conformidades, el comportamiento de los errores por tipo de artefacto.

Con este análisis se pretende enfocar el esfuerzo en las contribuciones más importantes, con objeto de optimizar el beneficio obtenido durante las pruebas.

En el paso 7 de la Figura 14: Señalar los elementos “pocos vitales” y los “muchos triviales” se visualiza que existe una frontera clara entre las dos categorías. En muchos casos no existe esta frontera claramente visible. En realidad se puede identificar generalmente una tercera categoría que J.M Juran llamó “zona dudosa”.

Una vez identificada esta tercera categoría en la gráfica se vuelve a realizar el análisis de Pareto en las nuevas condiciones y comprobar si los elementos incluidos en la anterior “zona dudosa” han pasado a ser “pocos vitales” y si su tratamiento es propicio. En el caso específico del ejemplo 2, los elementos pertenecientes a la “zona dudosa” pasan a ser parte de los “pocos vitales”, por eso en la interpretación de la gráfica anteriormente expuesta se tratan como tal.

En general, una vez tratados los elementos que claramente pertenecen a los “pocos vitales” se tiene un mejor conocimiento de lo que hay que hacer con los pertenecientes a la “zona dudosa”.

Esta herramienta de análisis de datos también se puede aplicar en el DPSW para identificar la tendencia de las no conformidades de un mismo artefacto en las diferentes iteraciones de prueba y realizar un análisis comparativo al final del proceso de prueba de liberación.

Anexo 13. Historial de pruebas exploratorias

Los aspectos a llenar en la tabla del historial de pruebas son los siguientes:

- Proyecto:
- Fecha de la reunión de inicio:
- Nombre del artefacto:
- Fecha en que se recibió el artefacto:
- Cantidad de páginas / CU / requisitos:
- Muestra seleccionada:
- Fecha en que se comenzó las PE:
- Fecha en que se terminó las PE:
- Tiempo total de ejecución de las PE/ horas – hombre:
- Cantidad de no conformidades por tipo:

- Fecha en que se notificó de la respuesta de no conformidades de las PE:
- Pruebas satisfactorias:
- Pruebas detenidas:
- Pruebas abortadas:
- Criterio de criticidad:

Anexo 14. Planilla de esfuerzo

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

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

Anexo 15. Selección de la muestra. Secciones de las listas de chequeo para revisar los artefactos

Criterios de selección. Artefactos de documentación

Para revisar los documentos especificación de requerimientos, especificación de CU y diseño de casos de prueba, se utilizarán de manera general las siguientes secciones de la lista de chequeo:

Estructura del documento. (Aplica a los tres artefactos)

- ¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?
- ¿Contiene las secciones obligatorias definidas en el expediente? (Ver expediente de proyecto definido por Calisoft).

Semántica del documento.

- ¿No ha identificado errores ortográficos?
- ¿Se entiende claramente lo que se ha especificado en el documento?
- ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?
- ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?

De manera específica para la especificación de requerimientos, se utilizará la siguiente sección de la lista de chequeo.

Elementos definidos por la metodología:

- ¿Están todos los requisitos redactados de forma simple y clara para aquellos que vayan a consultarlo en un futuro? Ver documento de especificación de requisitos.
- ¿Debería especificarse algún requisito con más detalle? Ver documento de especificación de requisitos.
- ¿Debería especificarse algún requisito con menos detalle? Ver documento de especificación de requisitos.
- ¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no cómo el sistema debe hacerlo? Ver documento de especificación de requisitos.

- ¿Han sido abordadas e identificadas los valores de entradas y salidas? Ver documento de especificación de requisitos.
- ¿Han sido incluidos las respuestas válidas y no válidas de los valores de entrada? Ver documento de especificación de requisitos.
- ¿Se puede verificar cada requisito? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requerimiento, ejemplo la especificación del caso de uso). Ver documento de especificación de requisitos.
- ¿Se han enumerado los requisitos incluso los que se derivan de otros requisitos? Ver documento de especificación de requisitos.
- ¿Se puede trazar cada requisito al origen en el entorno del problema, (caso de uso del negocio)? Ver documento de especificación de requisitos.
- ¿No aparece un mismo requisito en más de un lugar del documento de especificación? Ver documento de especificación de requisitos.
- ¿Son los requisitos consistentes? ¿No existe contradicción entre lo especificado por un requisito y lo especificado por otro? Ver documento de especificación de requisitos.
- ¿Satisface las especificaciones el nivel de detalle requerido por el equipo de diseño?
- ¿Los requerimientos se encuentran limpios de polarización y de implementación (no restringidos a una alternativa de diseño específica)?

De manera específica para la especificación de CU, se utilizará la siguiente sección de la lista de chequeo.

Elementos definidos por la metodología:

- ¿Cada caso de uso registra claramente lo que el sistema debe hacer?
- ¿Están clasificado los casos de uso que definen la arquitectura básica del sistema? (críticos)
- ¿Se ha descrito con precisión todas las alternativas o excepciones?
- ¿Está en infinitivo y refleja de manera clara el objetivo del usuario sobre el sistema?
- ¿El nombre del caso de uso es único?
- ¿El nombre del caso de uso es intuitivo?
- ¿El resumen dice como se inicia, como termina y las operaciones principales que realiza el caso de uso?
- ¿Se escribe una precondición si y solo si a partir de la ocurrencia de un suceso determinado comienza el caso de uso?
- ¿La precondición es válida tanto para flujos básicos como flujos alternativos?
- ¿La pos-condición plasma cambios que suceden en el sistema al terminarse de ejecutar el caso de uso?
- ¿Se especifica la complejidad del caso de uso?
- ¿Está descrito el caso de uso en presente?
- ¿Se describe de manera comprensible y detallada las acciones del actor frente al sistema? ¿Está lo más parecido a un manual de ayuda?
- ¿El caso de uso está relacionado con al menos un actor?
- ¿Si hay dos actores interactuando con el caso de uso está generalizado en uno solo?
- ¿Si el caso de uso es abstracto (include, extende, generalización-especialización), no lo inicializa ningún actor?
- ¿El flujo básico comienza diciendo “El caso de uso se inicia cuando el actor…”?
- ¿El flujo básico termina diciendo en un evento independiente “El caso de uso termina…”?
- ¿En el flujo básico no existen abreviaturas?
- ¿En el flujo básico las partes del flujo de eventos que se repiten en otro caso de uso se especifican como un caso de uso incluido?

- ¿En el flujo básico, si las alternativas que se describen casi nunca ocurren o son alternativas comunes a otros casos de uso se especifican como un caso de uso extendido?
- ¿En el flujo básico, si existe un proceso general y a partir de él se especializan otros, se especifican como una generalización / especialización?
- ¿Las alternativas o excepciones se reflejan como flujos alternos?
- ¿En todos los CU que se introducen datos tienen un flujo alterno donde el sistema valida la integridad de los datos que se introducen y muestra un mensaje en caso de que los datos estén incompletos?
- ¿Los flujos alternativos se nombran con el número del paso que lo generó en el flujo básico, una letra, ordenados alfabéticamente según quien lo produjo?
- ¿En la sección flujos alternativos se describen todas las excepciones que existan por muy evidentes que parezcan?
- ¿Al describir el caso de uso base se mencionan todos los casos de uso que extienden, se incluyen o se generalizan del caso de uso?
- ¿La descripción de los casos de uso incluidos, extendidos y especializados se realiza aparte?

De manera específica para el diseño de casos de prueba, se utilizará la siguiente sección de la lista de chequeo, si utiliza CU.

Elementos definidos por la metodología:

- ¿La descripción general del caso de prueba coincide con el resumen del caso de uso?
- ¿Se han especificado las condiciones iniciales que deben existir para que se realice el caso de prueba?
- ¿Se ha especificado la sección o las secciones que tiene el caso de uso? Las secciones, son flujos excluyentes dentro del caso de uso. Ejemplo: en el caso de uso Gestionar Libro las secciones serían eliminar, modificar y registrar un nuevo libro.
- ¿Se han especificado todos los escenarios de pruebas por cada sección? Un escenario es un "camino" completo a través del caso de uso. El flujo básico seria el que cumple con el objetivo del caso de uso y los flujos alternos serían las alternativas o excepciones.
- ¿Si el escenario tiene que ver con una entrada de datos, existe un escenario para validar que los datos sean correctos?
- ¿Si el escenario tiene que ver con una entrada de datos existe un escenario para verificar que no falten datos obligatorios?
- ¿Aparece una descripción de la funcionalidad por cada escenario?
- ¿El flujo del escenario especifica claramente el camino para probarlo?
- ¿Si existen variables se han definido sus nombres?
- ¿Si existen variables se han definido su clasificación (de acuerdo al tipo de dato)?
- ¿Si existen variables se ha definido si aceptan valores nulos o no?
- ¿Si existen variables se ha hecho una descripción de las mismas, especificando condiciones que debe cumplir?
- ¿Por cada escenario se han especificado las variables asociadas?
- ¿Por cada escenario se ha especificado el estado que las variable pueden tomar (válida, inválida, no aplica)?
- ¿Si el escenario implica que las variables tomen valor inválido, se han definido todas las posibles combinaciones, fijando una variable con el valor inválido y todas las demás con estado válido en cada combinación?
- ¿Se ha especificado la respuesta del sistema para cada escenario?
- ¿Durante la prueba se han colocado los valores que toman las variables en cada escenario?

De manera específica para el diseño de casos de prueba, se utilizará la siguiente sección de la lista de chequeo, si utiliza requerimientos

Elementos definidos por la metodología:

- ¿Se han especificado las condiciones que deben existir para que se realice el requisito?
- ¿Se ha especificado el nombre del requisito a probar?
- ¿Aparece una descripción de la acción que realiza el requisito?
- ¿Se han especificado todos los escenarios de pruebas asociados al requisito?
- ¿Si el requisito tiene que ver con una entrada de datos existe un escenario para validar que los datos sean correctos?
- ¿Si el requisito tiene que ver con una entrada de datos existe un escenario para verificar que no falten datos obligatorios?
- ¿El flujo del escenario especifica claramente el camino para probarlo?
- ¿Si existen variables se ha definido su nombre?
- ¿Si existen variables se ha definido su clasificación (de acuerdo al tipo de dato), si acepta valores nulos o no?
- ¿Si existen variables se ha definido si acepta valores nulos o no?
- ¿Si existen variables se ha hecho una descripción de la misma, especificando condiciones que debe cumplir la misma?
- ¿Por cada escenario se han especificado las variables asociadas?
- ¿Por cada escenario se ha especificado el estado que las variable pueden tomar (válida, inválida, no aplica)?
- ¿Si el escenario implica que las variables tomen valor inválido, se han definido todas las posibles combinaciones, fijando una variable con el valor inválido y todas las demás con estado válido en cada combinación?
- ¿Se ha especificado la respuesta del sistema para cada escenario?
- ¿Durante la prueba se han colocado los valores que toman las variables en cada escenario?

Lista de prioridades

Artefactos de documentación: documento de diseño y documento de arquitectura de información.

Características: Se seleccionará una cantidad de acápites al documento según Pareto, para ello se conforma una lista de prioridades, en dependencia de los criterios de los expertos.

Nota: Se revisarán sólo las páginas del documento que se seleccionen según la cantidad que define Pareto. Las páginas serán seleccionadas según una encuesta que se le realizó a varios expertos.

Validación según expertos: Se realizó una encuesta a 7 expertos, según la prioridad que ellos le asignaron a los diferentes acápites de los documentos antes mencionados, luego se hizo un promedio de los mismos y se le asignó un orden de prioridad a cada acápite de cada documento. A continuación se muestran las tablas con estos resultados.

Documento de diseño:

Tabla 23: Resultado ordenado de los acápites del documento de diseño.

Acápite	Promedio
Diagrama de clases	2
Descripción de clases	2,4

Diagrama de paquetes	3,6
Clase <<Nombre de la clase>>	3,8
Propósito	5,2
Introducción	5,5
Referencias	5,8
Alcance	6,4
Definiciones, acrónimos y abreviaturas	6,4

El 20 % de la cantidad total de acápites es 1.8, redondeando por exceso es aproximadamente 2. Por lo que la muestra significativa que se escogerá serán los acápites: Diagrama de clases y Descripción de clase (de acuerdo a los diagramas que seleccione de esas mismas serán las descripciones de clases que se van a revisar). Para revisar estos acápites se utilizarán las siguientes secciones de la lista de chequeo:

Estructura del documento:

- ¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?
- ¿Contiene las secciones obligatorias definidas en el expediente? (Ver expediente de proyecto definido por Calisoft)

Elementos definidos por la metodología:

- ¿Se ha especificado la visibilidad de los atributos, parámetros y las operaciones (pública, protegida, privada)?
- ¿Se han agregado a las clases de diseño todos los nombres de los métodos que aparecen en el diagrama de interacción de las clases de análisis?
- ¿Se ha indicado las relaciones de asociación, dependencia, generalización-especialización entre las clases de diseño?
- ¿Se ha indicado la navegabilidad de las asociaciones entre las clases?
- ¿Se ha indicado la cardinalidad entre las clases?
- ¿Son las clases de diseño una traza directa de las clases de análisis?
- ¿Cada operación de la clase de diseño está recogida en al menos una realización de caso de uso del análisis?
- ¿Las clases de diseño contienen los eventos o mensajes de las realizaciones de caso de uso del análisis?
- ¿De ser necesario se han creado otras clases para representar clases innatas del lenguaje de programación?
- ¿Si se usa una clase interface genérica del lenguaje se ha representado con el estereotipo interface?
- ¿Si existe una clase que implemente de la interface la relación, se ha representado como la de herencia pero con línea discontinua?
- ¿El nombre de cada operación de las clases de diseño es descriptible y “legible”?
- ¿Se ha descrito la interacción entre las clases de diseño?
- ¿Comienza el nombre de los eventos con un verbo?
- ¿Las instancias de los objetos siguen el orden de clase interfaz, clase controladora, clase entidad aunque no se represente con los estereotipos del análisis?

Semántica del documento:

- ¿Ha identificado errores ortográficos?
- ¿Se entiende claramente lo que se ha especificado en el documento?
- ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?

- ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?

Documento de arquitectura de información

Tabla 24: Resultado ordenado de los acápites del documento de arquitectura de información.

Acápite	Promedio
Diseño de la estructura de las pantallas tipo	2,4
Mapa de navegación	2,8
Descripción de los elementos que componen las pantallas	3,4
Elementos del sistema de navegación	3,8
Descripción de los elementos de la arquitectura	4,8
Diagrama de interacción	5
Esbozo de la estructura o taxonomía	7,4
Introducción	8,2
Alcance	8,4
Definiciones, acrónimos y abreviaturas	9,2
Referencias	9,4

El 20 % de la cantidad total de acápites es 2.2, redondeando por defecto es aproximadamente 2. Por lo que la muestra significativa que se escogerá serán los acápites: Diseño de la estructura de las pantallas tipo y mapa de navegación. Para revisar estos acápites se utilizarán las siguientes secciones de la lista de chequeo:

Estructura del documento:

- ¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?
- ¿Contiene las secciones obligatorias definidas en el expediente? (Ver expediente de proyecto definido por Calisoft)

Elementos definidos por la metodología:

- ¿Existe un mapa de navegación del producto?
- ¿El mapa de navegación representa las secciones, niveles y contenidos relacionados?
- ¿El sistema de navegación es consistente, uniforme y visible?
- ¿Se representa el diseño de la estructura de todas las pantallas tipo?
- ¿Se identifican las diferentes áreas en la estructura de las pantallas?

Semántica del documento:

- ¿Ha identificado errores ortográficos?
- ¿Se entiende claramente lo que se ha especificado en el documento?
- ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?
- ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?

Documento modelo de despliegue

Para revisar estos acápites se utilizarán las siguientes secciones de la lista de chequeo:

Estructura del documento:

- ¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?
- ¿Contiene las secciones obligatorias definidas en el expediente? (Ver expediente de proyecto definido por Calisoft).

Elementos definidos por la metodología:

- ¿Se han identificado los nodos y sus protocolos de comunicación?
- ¿Representa cada nodo un recurso de cómputo?
- ¿Se ha especificado claramente el nombre?
- De contener los nodos algunos componentes, ¿se han establecido las relaciones entre ellos (componentes)?
- ¿Se ha realizado la descripción de los nodos?
- ¿En la descripción se ha contemplado la capacidad de memoria y almacenamiento, sistema operativo, capacidad del micro u otras informaciones relacionadas con la capacidad?
- ¿Existe un nodo para una aplicación web? ¿Ha sido especificado?
- ¿Existe un nodo para el servidor de base de datos? ¿Ha sido especificado?
- ¿Los dispositivos están representados como un ortoedro?
- ¿Se ha especificado claramente el nombre?
- ¿Representa un recurso de cómputo que no tiene capacidad de almacenamiento?
- ¿Se ha tenido en cuenta el protocolo de comunicación para establecer las conexiones entre ellos?

Semántica del documento:

- ¿Ha identificado errores ortográficos?
- ¿Se entiende claramente lo que se ha especificado en el documento?
- ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?
- ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?

Documento glosario de términos

Para revisar estos acápites se utilizarán las siguientes secciones de la lista de chequeo:

Estructura del documento:

- ¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?
- ¿Contiene las secciones obligatorias definidas en el expediente? (Ver expediente de proyecto definido por Calisoft).

Elementos definidos por la metodología:

- ¿Todos los términos tienen una definición breve y clara?
- ¿Todos los términos del glosario están incluidos en alguna parte de las descripciones de los casos de uso del negocio?
- ¿Las definiciones de los términos no son generales y sí específicas para el proyecto?
- ¿Todas las palabras y expresiones que se utilizan para definir los términos están claras?

Semántica del documento:

- ¿No ha identificado errores ortográficos?
- ¿Se entiende claramente lo que se ha especificado en el documento?
- ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?
- ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?

Anexo 16. Confección de la lista de prioridades y validación según expertos

Tabla 25: Confección de la lista de prioridades para el documento modelo de diseño.

Acápites	Cantidad de NC	Total acumulativo	Porcentaje del total general	Porcentaje acumulativo
Diagrama de clases				
Descripción de clases				
Diagramas de paquetes				
Clase << Nombre de la Clase >>				
Propósito				
Introducción				
Referencias				
Alcance				
Definiciones, acrónimos y abreviaturas				

Tabla 26: Validación según experto. Documento modelo de diseño.

Acápites	Experto 1	Experto 2	Experto 3	Experto 4	Experto 5	Resultados
Introducción	7	6	3	7	6	5.5
Propósito	5	7	3	5	4	5.2
Alcance	6	8	3	9	7	6.4
Definiciones, acrónimos y abreviaturas	8	9	3	6	8	6.4
Referencias	9	1	3	8	9	5.8
Diagramas de paquetes	3	2	1	3	3	3.6
Diagrama de clases	1	3	2	1	1	2
Descripción de clases	2	4	3	2	2	2.4
Clase << Nombre de la Clase >>	4	5	3	4	5	3.8

Tabla 27: Confección de la lista de prioridades para el documento arquitectura de información.

Acápites	Cantidad de NC	Total acumulativo	Porcentaje del total general	Porcentaje acumulativo
Diseño de la estructura de las pantallas tipo				
Mapa de navegación				
Descripción de los elementos que componen las pantallas				
Elementos del sistema de navegación				
Descripción de los elementos de la arquitectura				
Diagrama de interacción				
Esbozo de la estructura o taxonomía				

Introducción				
Alcance				
Definiciones, acrónimos y abreviaturas				
Referencias				

Tabla 28: Validación según experto. Documento arquitectura de información.

Acápites	Experto 1	Experto 2	Experto 3	Experto 4	Experto 5	Resultados
Introducción	9	7	8	8	9	8.2
Alcance	8	8	8	10	8	8.4
Definiciones, acrónimos y abreviaturas	10	9	8	9	10	9.2
Referencias	11	10	8	7	11	9.4
Esbozo de la estructura o taxonomía	7	11	1	11	7	7.4
Descripción de los elementos de la arquitectura	5	6	2	6	5	4.8
Mapa de navegación	3	1	3	4	3	2.8
Elementos del sistema de navegación	4	2	4	5	4	3.8
Diseño de la estructura de las pantallas tipo	1	3	6	1	1	2.4
Descripción de los elementos que componen las pantallas	2	4	7	2	2	3.4
Diagrama de interacción	6	5	5	3	6	5

Tabla 29: Confección de la lista de prioridades para el documento modelo de despliegue.

Acápites	Cantidad de NC	Total acumulativo	Porcentaje del total general	Porcentaje acumulativo
Diagramas de despliegue				
Descripción de nodos				
Propósito				
Alcance				
Definiciones, acrónimos y abreviaturas				
Introducción				
Referencias				

Tabla 30: Validación según experto. Documento modelo de despliegue.

Acápites	Experto 1	Experto 2	Experto 3	Experto 4	Experto 5	Resultados
Introducción	4	6	3	5	4	4.4
Propósito	3	7	3	3	3	3.8
Alcance	5	1	3	7	5	4.2
Definiciones, acrónimos y abreviaturas	6	2	3	4	6	4.2

Referencias	7	3	3	6	7	5.2
Diagramas de despliegue	1	4	1	1	1	1.6
Descripción de nodos	2	5	2	2	2	2.6

Tabla 31: Confección de la lista de prioridades para el documento glosario de términos.

Acápites	Cantidad de NC	Total acumulativo	Porcentaje del total general	Porcentaje acumulativo
Definiciones				
Expediente de proyecto				
Casos de uso				
Lineamientos mínimos de calidad				
Casos de uso significativos para el negocio				
Prototipo				
Casos de uso arquitectónicamente significativos				
Pruebas de Liberación				
Pruebas de Aceptación				
No conformidades				
Propósito				
Solicitud de cambio				
Introducción				
Alcance				
Definiciones, acrónimos y abreviaturas				
Referencias				

Tabla 32: Validación según experto. Documento glosario de términos.

Acápites	Experto 1	Experto 2	Experto 3	Experto 4	Experto 5	Resultados
Introducción	13	4	11	15	13	11.2
Propósito	12	1	11	12	12	9.6
Alcance	14	2	11	16	14	11.4
Definiciones, acrónimos y abreviaturas	15	3	11	13	15	11.4
Referencias	16	4	11	14	16	12.2
Definiciones	1	5	11	1	1	3.8
Casos de uso	3	7	4	2	10	5.2
Pruebas de Aceptación	9	8	10	9	9	9
Expediente de proyecto	6	9	1	6	3	5
Pruebas de Liberación	8	10	7	10	8	8.6
Prototipo	5	11	6	8	4	6.8
Solicitud de cambio	11	12	9	11	5	9.6
No conformidades	10	13	8	7	7	9
Casos de uso significativos para el negocio	2	14	3	3	11	6.6
Casos de uso	4	15	5	4	6	6.8

arquitectónicamente significativos						
Lineamientos mínimos de calidad	6	16	2	5	2	6.4

Anexo 17. Mediciones al proceso. Métricas sobre el avance del proceso en la ejecución de las pruebas

Tabla 33: Métricas sobre el avance del proceso en la ejecución de las pruebas.

Elemento	Métrica	Propone medir
Funcionalidad	Cubrimiento de funcionalidades	Cantidad de casos de prueba ÷ Total de funcionalidades a probar.
	Tiempo por funcionalidad	Total de tiempo en ejecutar las pruebas ÷ Total de funcionalidades probadas.
	Densidad de defectos por funcionalidad	Total de No conformidades (NC) encontrados ÷ Total de funcionalidades probadas.
Casos de prueba o diseño de casos de prueba	Casos de prueba exitosos (1)	Contar cuantos casos de prueba fueron ejecutados y su resultado fue “Satisfactorio”.
	Casos de prueba que fallaron (2)	Contar cuantos casos de prueba fueron ejecutados y su resultado fue “Abortadas”.
	Casos de prueba que fallaron (2)	Contar cuantos casos de prueba fueron ejecutados y su resultado fue “Detenidas”.
	Casos de prueba ejecutados	Es la suma (1) y (2).
	Casos de prueba no ejecutados	Contar cuantos casos de prueba planificados para las PE no fueron ejecutados.
	Casos de prueba planificados	Contar cuantos casos de prueba se planificaron realizar en el ciclo de pruebas.
	Pruebas de regresión ejecutadas	Contar la cantidad de casos de prueba ejecutados en la 2da iteración, que ya fueron ejecutados en ciclos anteriores (PEI o 1ra iteración).
	Tiempo medio por caso de prueba	Tiempo total en ejecutar los casos de prueba ÷ Total de casos de prueba ejecutados.
	Densidad de las no conformidades (NC) por caso de prueba	Total de NC reportados en el ciclo de prueba ÷ Total de casos de prueba ejecutados.
	Eficiencia de las pruebas	Cantidad de pruebas ejecutadas ÷ total de NC encontradas.
Pruebas exploratorias	Esfuerzo de las PE	Contar el tiempo dedicado a las PE en el ciclo de pruebas.
	NC encontradas en las PE	Contar las NC encontradas durante las PE.
	Funcionalidades exploradas	Contar las funcionalidades exploradas durante las PE del ciclo de prueba.
No conformidades	NC por estado	Para cada estado de la NC. Contar las NC de ese estado en el ciclo de

		prueba.
	NC por tipo	Para cada tipo de NC (significativas, no significativas y recomendaciones) definida, contar la cantidad de NC reportadas en el ciclo de prueba.
	NC por clasificación	Para cada clasificación de NC, contar las NC según su clasificación en el ciclo de prueba.
	Tasa de reparación de NC	Cantidad de NC resueltas para este ciclo ÷ Cantidad de NC encontradas en el ciclo anterior (este aplica fundamentalmente para las PEP en la 2da iteración).
	Tiempo medio en resolver NC	Total de tiempo en resolver las NC desde que se reportaron ÷ Cantidad de NC resueltas en el ciclo de prueba.
	Tiempo medio en reportar NC	Total de tiempo en ejecutar casos de prueba ÷ Cantidad de NC reportadas en el ciclo de prueba.
Esfuerzo	Esfuerzo planificado (5)	Sumar el tiempo planificado para todas las actividades del proyecto de prueba.
	Esfuerzo realizado en el ciclo de prueba	Sumar el tiempo invertido en todas las actividades del ciclo de prueba.
	Esfuerzo realizado hasta el momento (3)	Sumar el tiempo realizado desde que el proyecto comenzó hasta el último ciclo realizado.
	Esfuerzo por realizar (4)	Sumar el tiempo que se planifica para cada actividad hasta que el proyecto de prueba termine.
	Esfuerzo total (6)	Sumar (3) y (4).
	Desviación en el esfuerzo planificado	Restar (6) menos (5).
	Esfuerzo por persona	Contar el esfuerzo por persona del equipo de prueba.
	Esfuerzo en reportar NC	Sumar el tiempo en reportar las NC encontrados en el ciclo de prueba.
Nota: Estas métricas igualmente se pueden adaptar para su utilización durante las PEP en la segunda iteración de las pruebas funcionales. Estados de las NC: nuevo, resuelta, pendiente, abortada, detenida, pospuesta, en curso, aceptada, no procede, asignada, rechazada, comentarios, liberado, aprobada, cerrada. Clasificación de las NC: funcionalidad, ortografía, formato, validación, redacción, error técnico, errores de interfaz, correspondencia con otra documentación, opciones que no funcionan, errores de idioma, excepciones, otros errores.		

Anexo 18. Modelo para la calidad interna y externa con las características y sub-características definidas por la ISO/IEC 9126-1 Parte 1: Modelo de Calidad

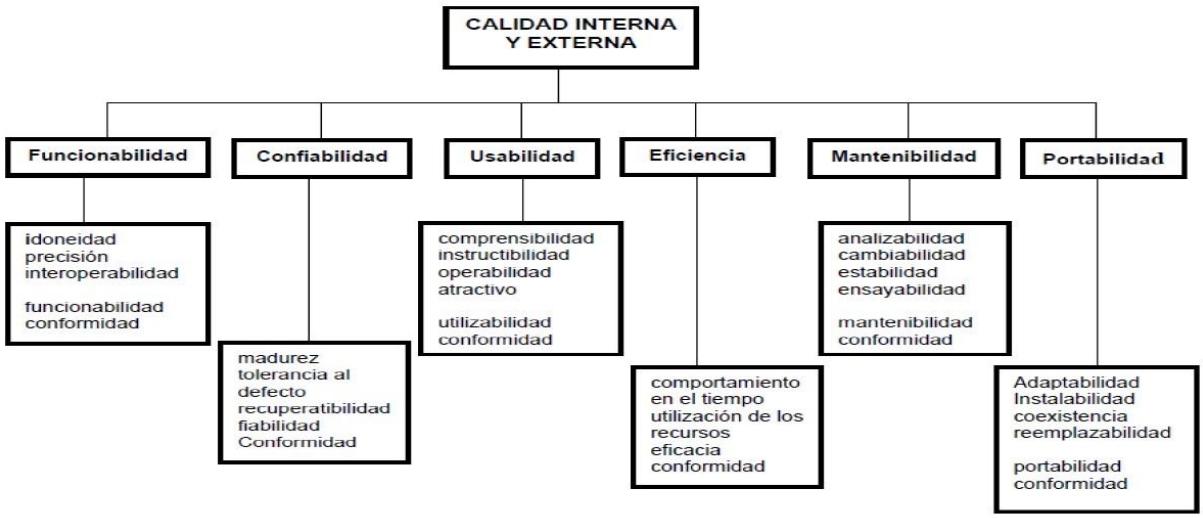


Figura 15: Modelo para la calidad interna y externa con las características y sub-características definidas por la ISO/IEC 9126-1 Parte 1: Modelo de Calidad.

Anexo 19. Métricas externas de calidad. Características de eficiencia, funcionalidad y confiabilidad. Descripción de las métricas

Tabla 34: Métricas externas de calidad. Características de eficiencia, funcionalidad y confiabilidad.

Característica Calidad	Elemento	Métrica	Propone medir
Eficiencia	Tiempo	Métrica # 26. Tiempo de respuesta	¿Cuál es el tiempo necesario para completar una tarea específica? ¿Cuánto tiempo se tarda antes de la respuesta del sistema a una operación especificada?
	Rendimiento	Métrica # 27 Rendimiento	¿Cómo muchas tareas pueden realizarse con éxito en un período determinado de tiempo?
	Eficacia	Métrica # 28: Cumplimiento de la eficiencia	¿Cómo es compatible la eficacia del producto a las normas, los estándares y regulaciones?
Funcionalidad	Idoneidad	Métrica # 4: Estabilidad en las especificaciones funcionales	¿Cuán estable son las especificaciones funcionales después de entrar a funcionar?
Confiabilidad	Madurez	Métrica # 12: Grado de solución ante fallos totales	¿Cuántas condiciones de fallo total están resueltas?
		Métrica # 14: Erradicación de fallos	¿Cuántos fallos han sido corregidos?

Descripción de las métricas de calidad

Métricas para la medición de la característica eficiencia.

- Métricas de eficiencia.

Tabla 35: Métrica externa de la sub-característica de calidad eficiencia.

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Tipo de medida
Cumplimiento de la eficiencia	¿Cómo es compatible la eficacia del producto a las normas, los estándares y regulaciones ?	Cuente el número de elementos que requieren de cumplimientos que se han cumplido y comparar con el número de elementos que requieran el cumplimiento según las especificaciones.	**X=1-A/B** (X- relación de elementos cumplimiento satisfechos en relación a la eficiencia) (A- Número de elementos cumplidos de la eficiencia que se especifica que no se han aplicado durante las pruebas). (B- número total de elementos cumplidos de la eficiencia especificado)	**0 < = X < = 1**	**X= Contable/contable** A= contable B= contable

- Métricas de comportamiento en el tiempo.

Tabla 36: Métrica externa de la sub-característica de calidad comportamiento en el tiempo.

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Tipo de medida
Tiempo de respuesta	¿Cuál es el tiempo necesario para completar una tarea	Iniciar una tarea específica. Medir el tiempo que tarda la muestra para	T= (tiempo al obtener el resultado) – (tiempo al entrar los datos)	**0 < T** Cuanto antes es mejor	**T = tiempo**

	específica? ¿Cuánto tiempo se tarde de la respuesta del sistema a una operación especificada ?	completar su operación. Mantener un registro de cada intento.			

- Métricas de rendimiento.

Tabla 37: Métrica externa de la sub-característica de calidad rendimiento.

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Tipo de medida
Rendimiento	¿Cómo muchas tareas pueden realizarse con éxito en un período determinado de tiempo?	Calibrar cada tarea de acuerdo a la prioridad dada por objeto. Iniciar las tareas de varios puestos de trabajo. Medir el tiempo que tarda la tarea en completar su operación. Llevar un registro electrónico de cada intento.	**X=A/T** A= número de tareas completadas T= período de tiempo de observación	**0 < X** El más grande es el mejor	**X = contable /tiempo** A= contable T= tiempo

Métricas para la medición de la característica funcionalidad.

- Métrica de idoneidad.

Tabla 38: Métrica externa de la sub-característica de calidad idoneidad.

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Tipo de medida
Estabilidad en las especificaciones funcionales	¿Cuán estable son las especificaciones funcionales después de entrar a funcionar ?	Cuenta el número de funciones que se describen en las especificaciones funcionales que tuvo que ser cambiado	**X=1-A/B** A- Número de funcionalidades actualizadas después de	**0<=X<=1** A mayor cercanía al 1 resultará mejor.	**X=Contable / Contable** A=Contable B=Contable

		después de que el sistema se pone en funcionamiento y compararlo con el número total de funciones que se describen en las especificaciones de requisitos.	entrar en funcionamiento. B- Número de funcionalidades descritas en la especificación de requisitos.		

Métricas para la medición de la característica confiabilidad.

- Métrica de madurez.

Tabla 39: Métrica externa de la sub-característica de calidad madurez.

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Tipo de medida
Grado de solución ante fallos totales	¿Cuántas condiciones de fallo total están resueltas?	Cuente el número de fallos totales que no se repitieron en determinado período de pruebas bajo condiciones similares. Mantenga un reporte de solución de problemas describiendo la situación de todos los fallos totales.	**X = A1 / A2** A1 - Número de fallos totales solucionados. A2 - Número total de problemas Reales detectados.	**0 <= X <= 1** A mayor cercanía al 1 resultará mejor, cuantos más fallos totales estén resueltos.	**X = Contable/ Contable** A1 = Contable A2 = Contable
Erradicación de fallos	¿Cuántos fallos han sido corregidos?	Cuente el número de fallos resueltos durante el período de pruebas y compárelos con el número total de fallos detectados y el número total de fallos pronosticados.	**1) X = A1 / A2** A1 - Número de fallos solucionados. A2 - Número total de fallos reales detectados. **2) Y = A1 /A3** A3 - Número total de fallos latentes pronosticados.	**0 <= X <= 1** A mayor cercanía al 1 resultará mejor (cuanto menos fallos queden) **0 <= Y** A mayor cercanía al 0 resultará mejor (cuanto menos fallos queden)	**1) X =Contable/ Contable** A1 = Contable A2 = Contable **2) Y= Contable/ Contable** A3 = Contable

Anexo 20. Comparación entre la cantidad de no conformidades detectadas durante la ejecución de las pruebas exploratorias iniciales y la primera iteración de prueba

Tabla 40: Comparación entre la cantidad de no conformidades detectadas durante la ejecución de las pruebas exploratorias iniciales y la primera iteración de prueba. Descripción de las NC por tipo de error.

En las PEI				Sin ejecutar PE (1ra it)				Después de las PE (1ra it)				Resultados
Proyecto	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Cant. NC**	**Proyecto**	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Cant. NC**	**Proyecto**	**Artefacto**	**Cant. pág, CP, DCP o requisitos**	**Cant. NC**	**NC**
A	Manual de usuario	28 pág.	Ortografía (2), error técnico (1), formato (11), redacción (6), TOTAL (20)	B	Manual de usuario	16 pág.	Formato (6), ortografía (2), redacción (14), otros (1), recomendación (2), TOTAL (25)	A	Manual de usuario	28 pág.	Correspondencia con ortro artefacto (6), error técnico (3), formato (4), redacción (4), recomendación (1), TOTAL (18)	**7 NC**
C	Sistema	45 CP	Funcionalidad (2), excepciones (1), validación (1), redacción (2), TOTAL (6)	D	Sistema	41 DCP	Error técnico (1), correspondencia con otro artefacto (6), validación (28), opciones que no funcionan (2), interfaz (13),	C	Sistema	45 CP	Correspondencia con ortro artefacto (15), error interfaz (12), excepciones (3), formato (3), funcionalidad (3), ortografía (7), validación (36), otros (1), recomendación (2), redacción (6), error	**4 NC**

							funcionalidad (34), otros (8), recomendación (2), TOTAL (94)				idioma (1), diseño (1), TOTAL (90)	
E	Sistema	145 CP, (29 CP muestra)	Funcionalidad (2), validación (17), ortografía (1), redacción (3),errores de interfaz (4), excepciones (1), correspondencia con otro artefacto (11), otros (3), TOTAL (42)	F	Sistema	156 CP	Correspondencia con otro artefacto (4), ortografía (12), redacción (4), validación (20), opciones que no funcionan (7), errores de interfaz (2), funcionalidad (49), excepciones (9), otros (5), diseño de CP (3), TOTAL (115)	E	Sistema	145 CP	Formato (2), error técnico (12), correspondencia con otro artefacto (12), ortografía (6), redacción (11), validación (4), funcionalidad (4), otros (10), recomendación (1), TOTAL (62)	**53 NC**
G	Sistema	9 CP (muestra 2 CP)	Funcionalidad (5), ortografía (1), error de interfaz	H	Sistema	14 CP	Correspondencia con otro artefacto (1),	G	Sistema	9 CP	Validación (1), recomendación (4), TOTAL (5)	**37 NC**

			(5), validación (6), correspondencia con otro artefacto (1) y redacción (3), TOTAL (21)				ortografía (5), redacción (5), validación (10), opciones que no funcionan (1), error de idioma (1), funcionalidad (10), recomendación (9), TOTAL (42)					
I	Sistema	4 CP (muestra los escenarios más complejos)	Funcionalidad (1), ortografía (4), redacción (1) TOTAL (6)	J	Sistema	6 CP	Ortografía (2), correspondencia con otro artefacto (1), redacción (4), validación (11), funcionalidad (3), error de interfaz (1), opciones que no funcionan (1), otro	I	Sistema	4 CP	Correspondencia con otro artefacto (1), ortografía (1), error de interfaz (1), excepciones (1), recomendaciones (1), TOTAL (5)	**19 NC**

							(1), TOTAL (24)					
K	Sistema	23 CP (6 CP muestra)	Correspondencia con otro artefacto (4), redacción (1), validación (1), error interfaz (3), funcionalidad (2), diseño CP (1), recomendación (2) TOTAL (14)	L	Sistema	29 CP	Correspondencia con otro artefacto (15), validación (15),interfaz (8), otros (4), funcionalidad (6), TOTAL (48)	K	Portal web	23 CP	Ortografía (3), redacción (4), validación (2), error interfaz (2), error idioma (2), diseño CP (16), recomendación (3), TOTAL (32)	**16 NC**
M	Portal web	73 CP (19 CP de muestra)	Formato (4), redacción (1), TOTAL (5)	N	Sistema	65 DCP	Validación (15), funcionalidad (92), excepciones (9), otros (18), ortografía (4), opciones que no funcionan (14), error interfaz (7), redacción (2),	M	Portal web	73 CP	Correspondencia con otro artefacto (4), error interfaz (1), excepciones (1), funcionalidad (8), opciones que no funcionan (1), otros (2), recomendación (1), validación (2), TOTAL (20)	**151 NC**

							correspondencia con otro artefacto (3), error idioma (2), formato (3), error técnico (2), TOTAL (171)					
O	Portal web	23 CP (5 DCP muestra)	Correspondencia con ortro artefacto (4), redacción (1), validación (1), error interfaz (3), funcionalidad (2), diseño CP (1), recomendación (2) TOTAL (14)	P	Sistema	21 DCP	Funcionalidad (60), formato (20), error técnico (4), validación (1), ortografía (20), recomendación (1), TOTAL (106)	O	Portal web	23 CP	Ortografía (3), redacción (4), validación (2), error interfaz (2), error idioma (2), diseño CP (16), recomendación (3), TOTAL (32)	**74 NC**
Q	Sistema	4 CP (muestra los escenarios más complejos)	Funcionalidad (1), ortografía (4), redacción (1) TOTAL (6)	R	Sistema	6 CP	Ortografía (2), correspondencia con otro artefacto (1), redacción (4),	Q	Sistema	4 CP	Correspondencia con otro artefacto (1), ortografía (1), error de interfaz (1), excepciones (1), recomendacion	**37 NC**

							validación (11), funcionalidad (3), error de interfaz (1), opciones que no funcionan (1), otro (1), TOTAL (24)				es (1), TOTAL (5)	
S	Sistema	7 CP (2 CP de muestra)	Diseño de CP (3)	T	Sistema	6 CP	Ortografía (1), redacción (9), validación (16), funcionalidad (3), error de interfaz (4), excepciones (1), TOTAL (34)	S	Sistema	7 CP	Funcionalidad (2), error de idioma (2), validación (1), redacción (1), TOTAL (6)	**28 NC**
U	Sistema	63 CP (16 CP de muestra)	Ortografía (7), redacción (6), validación (4), error de interfaz (1), recomenda	V	Sistema	45 CP	Formato (3), ortografía (1), validación (3), opc. que no funcionan (2), error	U	Sistema	63 CP	Correspondencia con otro artefacto (6), formato (1), ortografía (7), redacción (3), validación (4), opciones que no funcionan	**11 NC**

			ciones (8), TOTAL (26)				de interfaz 2, funcionalidad (11), excepciones (6), otros (5), diseño del CP (7), recomendación (1), redacción (1), TOTAL (44)				(1), error de interfaz (6), funcionalidad (2), otros (1), recomendaciones (2), TOTAL (33)	
W	Sistema	45 CP	Funcionalidad (2), excepciones (1), validación (1), redacción (2), TOTAL (6)	X	Sistema	54 CP	Error técnico (2), redacción (3), diseño (16), error de idioma (1), validación (19), interfaz (7), funcionalidad (32), recomendación (17), TOTAL (97)	W	Sistema	45 CP	Correspondencia con ortro artefacto (15), error interfaz (12), excepciones (3), formato (3), funcionlidad (3), ortografía (7), validación (36), otros (1), recomendación (2), redacción (6), error idioma (1), diseño (1), TOTAL (90)	**7 NC**

Anexo 21. Recolección de datos de los proyectos y cálculo de las métricas

Los datos son recogidos mediante la herramienta para la gestión de las métricas propuesta en (Góngora 2012), en las tablas a continuación se muestra un ejemplo de la plantilla utilizada para la recogida de los datos. Cada especialista al frente del proyecto que se encuentre en pruebas será el encargado de registrar los datos para posteriormente proceder al cálculo de la métrica de calidad. Los datos son verificados en cuanto a exactitud y a la unidad de medida apropiada.

A continuación se muestran los valores que se les asignó a las variables de las métricas según la recolección que se obtuvo de los datos de los proyectos y los resultados del cálculo de las métricas.

El primer ejemplo que se presenta corresponde a los proyectos E y F de la Tabla 9: Comparación entre el tiempo de ejecución de las pruebas de proyectos que se le aplicaron PE y proyectos que no. Al proyecto E se le realizaron PEI y al proyecto F no.

Tabla 41: Variables de la métrica externa de la sub-característica de calidad rendimiento.

Nombre de la métrica	Variables	Datos proyecto E– 1ra iteración con PEI	Datos proyecto F – 1ra iteración sin PEI
Rendimiento	A – Número de tarea completada. B – Período de tiempo de observación.	A = 9 B = 2	A = 14 B = 14

Una vez recopilados todos los datos necesarios de los proyectos seleccionados se procede a calcular las métricas. Cada métrica de calidad tiene una fórmula y una interpretación diferente de los resultados que arroja. El cálculo de las métricas se realiza mediante la herramienta para la gestión de las métricas propuesta en (Góngora 2012), la misma utiliza una plantilla con los datos que se muestran en la Tabla 42.

Tabla 42: Resultados del cálculo de la métrica externa de la sub-característica de calidad rendimiento.

Proyecto	Artefacto	Iteración	Sub-característica	Métricas	Nivel requerido	Valor
Proyecto E	Aplicación	1ra iteración con PEI	Rendimiento	X=A/T	0<X El más grande es el mejor	4.5
Proyecto F	Aplicación	1ra iteración sin PEI	Rendimiento	X=A/T	0<X El más grande es el mejor	1

El segundo ejemplo que se presenta corresponde a los proyectos E y F de la Tabla 8: Comparación entre la cantidad de no conformidades detectadas durante la ejecución de las PEI y la primera iteración de prueba. Los datos del proyecto E se obtuvieron después de habérsele realizado PEI y los del proyecto F una vez culminada la primera iteración de las pruebas funcionales sin haberle realizado PEI.

Tabla 43: Variables de la métrica externa de la sub-característica de calidad idoneidad.

Nombre de la métrica	Variables	Datos proyecto E– 1ra iteración con PEI	Datos proyecto F – 1ra iteración sin PEI
Adecuación funcional	A – Número de problemas detectados en las funcionalidades del sistema. B – Número de casos de prueba evaluados.	A = 62 B = 145	A = 115 B = 156

A continuación en la Tabla 44 se presenta el cálculo de la métrica adecuación funcional perteneciente a la sub-característica de calidad de idoneidad de los proyectos E y F respectivamente.

Tabla 44: Resultados del cálculo de la métrica externa de la sub-característica de calidad idoneidad.

Proyecto	Artefacto	Iteración	Sub-característica	Métricas	Nivel requerido	Valor
Proyecto E	Aplicación	1ra iteración con PEI	Idoneidad (adecuación funcional)	X=1-A/B	1 (0<=X<=1)	0.6
Proyecto F	Aplicación	1ra iteración sin PEI	Idoneidad (adecuación funcional)	X=1-A/B	1 (0<=X<=1)	0.3

Anexo 22. Diseño de la encuesta # 2

Cuestionario

Con este cuestionario se pretende identificar las potencialidades que brindan la aplicación de la estrategia y el nivel de cumplimiento de la situación problémica y el problema. Le pedimos sinceridad a la hora de responder las preguntas, le aseguramos confidencialidad y anonimato a su respuesta, solamente debe mencionar el rol que desempeña o su responsabilidad en la universidad.

Rol: _____

Responsabilidad: _____

Responda las siguientes preguntas y marcar con una x en el caso que corresponda:

1. Sobre el nivel de importancia, necesidad y utilidad de la estrategia de Pruebas Exploratorias, que valor usted le daría en una escala del 1-10:

   Importancia:

1	2	3	4	5	6	7	8	9	10

   Necesidad:

1	2	3	4	5	6	7	8	9	10

   Utilidad:

1	2	3	4	5	6	7	8	9	10

2. Considera usted que con la aplicación de las Pruebas Exploratorias en el Laboratorio Industrial de Pruebas de Software se ahorre:

Tiempo: Sí ___ No ___ No sé ___
Esfuerzo: Sí ___ No ___ No sé ___
Recursos: Sí ___ No ___ No sé ___
Otros: Cuáles: _____

3. Considera usted que con la aplicación de las Pruebas Exploratorias en el proceso de Pruebas de Liberación se logre mayor:

   Eficiencia del laboratorio: Sí ___ No ___ No sé ___
   Rendimiento del laboratorio: Sí ___ No ___ No sé ___

   Fundamente su respuesta: _____
   _____

   a. ¿Qué otros aspectos se deben tener en cuenta para potenciar estos indicadores?

   _____
   _____

4. Las Pruebas Exploratorias Iniciales se aplicarán antes de la 1ra iteración de prueba, sin embargo, durante el proceso de Pruebas de Liberación también puede realizarse una exploración del producto (Pruebas Exploratorias), considera usted que estás pueden ejecutarse paralelas a la:
   1ra iteración: ____
   2da iteración: ____
   3ra iteración: ____
   No sé: ____
   Nunca: ____ ¿Por qué?_____

5. Si existieran los Laboratorios de Pruebas de Software en pequeñas sedes diseminadas por la geografía del país, considera usted que se puedan ejecutar en ellas las Pruebas Exploratorias:

   Sí ___ No ___ No sé ___
   Por qué (marque todas las que considere):
   □ Se pueden realizar con pequeños equipos.
   □ Utiliza pocos recursos.
   □ Durante su ejecución no es necesario realizar pruebas especializadas.
   □ El tiempo empleado en su ejecución es menor que el de una iteración completa de pruebas.
   □ El cliente puede obtener una evaluación de la calidad del producto en breve plazo.
   □ Otros, fundamente su respuesta:
   _____
   _____

Son de gran interés las respuestas que ha proporcionado. Muchas gracias.

GLOSARIO DE TÉRMINOS

Actividad: Conjunto de operaciones o tareas propias de una persona o entidad que permite que el trabajo ha realizar sea descrito y entendido de manera precisa por aquellos que tienen que ejecutarlo.

Artefacto: Un artefacto es todo tipo de información creada, producida, cambiada o usada por el proceso, es un producto del proceso. Los artefactos pueden tener asociado una plantilla, como guía de su contenido. Los artefactos podrán asociarse a las actividades como entrada o salida de las mismas.

Calidad: Conjunto de propiedades y características de un producto o servicio que le confieren su aptitud para satisfacer unas necesidades explícitas o implícitas.

Calidad de software: La concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente.

Ciclo de prueba: Es la ejecución del proceso de prueba contra una versión identificada del producto a probar.

Eficacia: mide los resultados alcanzados en función de los objetivos que se han propuesto, presuponiendo que esos objetivos se mantienen alineados con la visión que se ha definido. Se encuentra en el equilibrio entre la producción de los resultados deseados y la capacidad de producción.

Eficiencia: Capacidad de disponer de alguien o de algo para conseguir un efecto determinado.

Equipo de desarrollo: Es un grupo de trabajo constituido por una serie de profesores, investigadores, colaboradores y alumnos unidos en la ilusión de acometer un determinado proyecto o avanzar en el conocimiento y en la investigación teórica y aplicada.

Fases e Iteraciones: Los procesos de desarrollo de software dividen el desarrollo en ciclos donde cada ciclo trabaja para construir una nueva generación del sistema y a su vez cada ciclo se divide en fases y las fases en iteraciones. Cada fase tiene un conjunto de objetivos definidos y solamente se puede dar por concluida al alcanzar estos objetivos. Una iteración es una recorrida completa a través de las disciplinas, realizando una mini-cascada en las actividades definidas, resultando en una liberación (interna o externa) de un ejecutable, el cual es un subconjunto del sistema final que crece incrementalmente de iteración en iteración hasta llegar al sistema final. Se permitirá la creación, modificación, borrado y consulta de fases e iteraciones

Herramientas: Utensilios o provisiones necesarias para poder emprender un proyecto de software. Soportan los procesos de desarrollo de software modernos. Las herramientas asisten en la realización de las diversas actividades del proceso. Las herramientas podrán ser asociadas o desasociadas a una actividad. Esto indica que para realizar la actividad se usa la herramienta. Las mismas pueden ser automatizadas o no.

No conformidad: De acuerdo a la definición en la norma NC ISO 9000: 2005 (3.1.2), una no conformidad es el incumplimiento de un requisito.

Métricas: una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Proceso: Es un conjunto de actividades y resultados asociados que producen un resultado.

Producto: Conjunto de artefactos que se crean durante la vida del proyecto, como los modelos, código fuente, ejecutables y documentación.

Proyecto: Combinación de recursos humanos y no humanos reunidos en una organización temporal para conseguir un propósito, tiene un punto de de comienzo definido y con objetivos definidos mediante los que se identifican.

Proyecto de software: El elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión de un producto.

Rendimiento: Producto o utilidad que rinde de algo o de alguien y proporción entre el producto o el resultado obtenido y los medios utilizados.

SIGLAS

BPO: Business Process Outsourcing.

CMMI: Capability Maturity Model Integration.

ERP: Enterprise Resource Planning.

ISTQB: International Software Testing Qualifications Board.

Pymes: Pequeñas y Medianas Empresas.

SAP: NetWeaver Application Server.

SEI: Software Engineering Institute.

SQA: Software Quality Assurance.

SLA: Service Level Agreement.