Universidad de las Ciencias Informáticas

Facultad 6

Título:

"Desarrollo de las Pruebas de Liberación del Sistema de Ayuda Médica para la atención a las Dislipoproteinemias"

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autores: Yunior Mesa Reyes

Yuniel Torres Rodríguez

Tutores: Ing. Lázaro Cánova Amador

Ing. Ileane Martí Pérez

Co-Tutor: Ing. Heney Díaz Pérez

Ciudad de la Habana

Junio, 2009

El conocimiento nos hace Responsables.

Che

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2009.

AUTORES

_____ _____

Yunior Mesa Reyes Yuniel Torres Rodríguez

TUTORES

_____ _____

Ing. Lázaro Cánova Amador Ing. Ileane Martí Pérez

CO - TUTOR

_____

Ing. Heney Díaz Pérez

DATOS DE CONTACTO

Tutores:

- ✓ Ing. Lázaro Cánova Amador

Ingeniero Informático, CUJAE 2005. Profesor de la asignatura de Teleinformática. Posee categoría docente de Instructor. Universidad de las Ciencia Informáticas, La Habana, Cuba.

e-mail: lcanova@uci.cu.

- ✓ Ing. Ileane Martí Pérez

Universidad de las Ciencia Informáticas (UCI), La Habana, Cuba. Ingeniera en Ciencias Informáticas. Profesora de la asignatura de Teleinformática. Posee categoría docente de Instructor Recién Graduado. Actualmente se desempeña como profesora de Teleinformática Jefa del Departamento del Polo de Gestión de Información Biomédica, en la Facultad 6 de la UCI.

e-mail: imarti@uci.cu.

Co-Tutor:

- ✓ Ing. Heney Díaz Pérez

Universidad de las Ciencia Informáticas, La Habana, Cuba. Ingeniero en Ciencias Informáticas. Especialista de la Dirección de Calidad de Software de la Infraestructura Productiva, grupo Laboratorio de Pruebas.

e-mail: hdperez@uci.cu.

Agradecimientos.

"Sentir gratitud y no expresarla es como envolver un regalo y no darlo."
William Arthur Ward

YUNIOR MESA REYES

Agradezco a todas las personas (familiares, profesores y amigos) que de una forma u otra contribuyeron con mi formación, especialmente:

A mis padres porque siempre me han ayudado en todo lo que estuvo a su alcance, por haber soportado mis malacrianzas, por haberme apoyado siempre y lo más importante, haber confiado en que nuestro sueño de tantos años se está realizando. A mi tía Estrella por soportarme durante todo este tiempo.

A todos los profesores que de una forma u otra influyeron en mi formación durante estos 5 años y en especial al profe Edistio que sin su perseverancia y paciencia para conmigo no hubiera sido posible ni tan si quiera haber comenzado este trabajo de diploma.

A mi dúo de tesis y a mis tutores por sus sugerencias siempre bien recibidas y haberme guiado en esta etapa final que han dado como resultado este trabajo de diploma y mi graduación como ingeniero.

A Heney, por demostrarme que la incondicionalidad aun existe, por ayudarme y apoyarme incondicionalmente. Sin tu ayuda y apoyo todo me hubiera resultado mucho más difícil. No tengo palabras para agradecer todo lo que has hecho por mí.

Al piquete del grupo de calidad de la Facultad, especialmente los de 5to año por su ayuda incondicional en todo momento.

A todos los muchachos y muchachitas del comité por los momentos buenos y malos que compartimos y a la profe Berta que verdaderamente ha sido una escuela de la vida para mí, por sus consejos y enseñanzas en cada frente de batalla.

A todos una vez más…..…… Muchas Gracias

YUNIEL TORRES RODRÍGUEZ

Ha llegado el momento de agradecer a todos los que de una forma u otra hicieron posibles que llegara hasta aquí especialmente:

A mis padres Alina y Jorge por todo el amor y el apoyo que siempre me han brindado, por su paciencia, dedicación, comprensión y por mantener siempre unida esta familia, por ser los principales responsables de que yo haya cumplido este gran sueño, dos seres a los cuales agradezco por existir, por confiar en mí y por estar a mi lado justo en el momento y lugar donde los necesito, y a los que toda una vida no me alcanzaría para recompensarlos y agradecerles porque todo cuanto soy y tengo es gracias a ellos Gracias por enseñarme el camino correcto, sin ustedes, este momento no hubiese sido posible.

A Dios por estar siempre conmigo ayudándome y fortaleciéndome en los momentos más difíciles de mi vida y por permitirme decir Ebenezer.

A Erick mi hermanito lindo, el cual es parte de mi vida y por el cual me he esforzado para realizar este sueño.

A mi Madrastra, a mi Padrastro, a mis tías, tíos, primos, familiares y amigos en especial a Julia, Villa, Yamilet, Papo, Josefina y Flora por su amor, cariño, ternura y dedicación , además por ayudarme y apoyarme en todo, en estos 5 años.

A mi dúo de tesis, tutores y profes que me han ayudado a lo largo de estos inolvidables 5 años y especialmente al profe Heney.

A Veylys por ser la iniciadora de este Trabajo de Diploma conmigo.

A todos una vez más…..…… Muchas Gracias.

Agradecimientos Especiales…

Agradecemos infinitamente a nuestra Revolución, que nos haya permitido convertirnos en lo que hoy somos y que haya puesto en nuestras manos la honrosa tarea de llevar hacia delante este país, por eso y más, este trabajo de diploma constituirá el primer paso de avance hacia ese camino.

A nuestro Comandante en jefe, Fidel Castro Ruz, por su incalculable capacidad de dirigir, su carácter intransigente y su inquebrantable confianza en la tropa del futuro, gracias.

Dedicatoria.

A mis familiares, profesores y amistades.

A mis dos tesoros más preciados, mis padres María y Nelson, mi hermanita Yisley, mi abuelita Zoila y mi tía La China que también forma parte de este sueño, mi pequeña familia de oro, como yo les llamo.

De forma general a mis familiares, amigos, profesores y todas las personas que de una forma u otra estuvieron siempre cerca durante estos cinco años…

Yunior.

A mis padres Alina y Jorge, por su confianza y apoyo incondicional, por quererme tanto, por ser el motivo de mi felicidad y la inspiración para este trabajo.

A Erick mi hermanito lindo.

A familiares, amigos, profesores y todas las personas que de una forma u otra estuvieron siempre cerca durante estos cinco años…

Yuniel.

Resumen.

La Propuesta de las Pruebas de Liberación del Sistema de Ayuda Médica para la atención a las Dislipoproteinemias (alasLIPO) está orientada a identificar el proceso de liberación del software establecido, determinando las técnicas que se realizaran y las herramientas utilizadas, con el rol involucrado en la realización de las pruebas.

Para lograr la liberación del sistema se puso en práctica la búsqueda de la información necesaria; centrada fundamentalmente, en la confección del Plan de Pruebas y la aplicación de las mismas.

Se realizó la liberación de la Documentación y Aplicación emitiendo las No Conformidades detectadas durante el proceso de liberación así como Recomendaciones al grupo de desarrolladores, además se realizaron las Pruebas de Carga y Estrés para evaluar el comportamiento del sistema a partir de condiciones anormales, emitiendo posteriormente una valoración general del proceso realizado.

PALABRAS CLAVES

Calidad de Software, Casos de Prueba, Caja Negra, Carga y Estrés, Plan de Prueba, Pruebas.

ÍNDICE

INDICE DE FIGURAS

INTRODUCCIÓN

La Calidad del Software consiste en desarrollar productos lógicos que, cumpliendo con las normas establecidas, satisfagan las necesidades del usuario, los requisitos implícitos, y que tiendan a cero defectos, siendo todavía hoy en día una disciplina minoritaria.

El software se ha convertido en un tema crítico en la sociedad moderna mundial. Todos parecen necesitar mejores software en el menor tiempo posible y a menor costo.

Existe una tendencia internacional a la producción de software avalado por sistemas de calidad, pero esta práctica se puede decir que no está generalizada en todos los países.

Una característica esencial del software de calidad es su capacidad para ser reutilizado. El problema de ámbito universal más caro de la historia tiene que ver con el software, con la falta de calidad y con la imposibilidad de la reutilización del software por su incapacidad de adaptarse a circunstancias que sobrevienen de manera inevitable.

En los últimos años la empresa del software ha tomado gran auge a nivel mundial, de modo que nuestro país ha querido incursionar de manera más estable en este mundo del desarrollo de software con la gran idea de que en un futuro se cuente en el país con la gran Industria Cubana de Software.

El desarrollo de la economía cubana en los momentos actuales está vinculado a los avances relacionados con las nuevas tecnologías de la información y las comunicaciones.

La producción de software se convierte en una actividad cada vez más demandada, de ahí surge la necesidad de garantizar que los clientes se sientan a gusto con un software de alta calidad.

El software se puede crear de forma genérica o a la medida y necesidad del cliente, la mayoría con características contables y económicas. El país lucha por cumplir estrictamente con las necesidades y requerimientos del cliente, sin dejar a un lado que el software debe ser producido con la calidad requerida y liberado en el menor tiempo posible.

La inclinación hacia el producto se debe al cumplimiento de los distintos atributos de la calidad, estándares, requisitos y las características exclusivas del software que se han seguido en el transcurso del desarrollo del mismo. Por este motivo, si no se controla la calidad, no se puede conocer si se va incrementando gradualmente la madurez en el proceso.

Obtener buena calidad en el desarrollo del software es uno de los retos más difíciles de enfrentar; un programa debe operar correctamente o proporcionará poco valor a sus usuarios, de ahí la necesidad de probar el sistema. La promoción de la industria cubana del software ha tenido como línea estratégica aprovechar la enorme credibilidad que tiene nuestro país en sectores tan vitales como la salud, la educación y el deporte.

La producción sostenida de software de alta calidad, tiene una positiva repercusión en el incremento de la exportación e importación. La Facultad 6, con la conformación de los perfiles de Bioinformática y Gestión de Información Biomédica (GIB), ha dado lugar a la realización de proyectos relacionados con la rama de la medicina cubana, tal es el caso del Sistema de Ayuda Médica para la atención a las Dislipoproteinemias (alasLIPO).

El correcto funcionamiento del sistema es una de las premisas fundamentales en este proceso de pruebas, siendo este un software que se implantará en todo el sistema de salud cubana que permitirá obtener el diagnóstico y tratamiento a pacientes con enfermedades dislipidémicas, por ello este trabajo está encaminado a una revisión minuciosa y a la realización de las pruebas que darán como resultado el estado real en que se encuentra.

Por lo anteriormente expuesto surge el **problema científico** que guía a la realización del trabajo de diploma, ¿Cómo garantizar que alasLIPO cumpla con los requisitos y la calidad requerida para su posterior implantación? Teniendo como **objeto de estudio** el proceso de pruebas de calidad de software y enmarcando la investigación en el **campo de acción** del proceso de pruebas de Liberación en la Facultad 6.

Dada la problemática planteada y con el propósito de dar solución al problema propuesto se plantea como **objetivo general** desarrollar las pruebas de liberación de la documentación y la aplicación de alasLIPO. Contando con los **objetivos específicos** siguientes:

- Realizar la revisión técnica a la documentación del producto.
- Diseñar las pruebas a aplicar al sistema.
- Aplicar las pruebas diseñadas.
- Evaluar los resultados obtenidos.

Entre las **tareas de la investigación científica** que se propone desarrollar para el cumplimiento de los objetivos se señalan:

- Investigación de los tipos de pruebas que pueden ser aplicadas, como funcionan y los pasos a seguir su correcta puesta en práctica y así certificar la calidad de un software.
- Estudio de las actividades propuestas por la metodología de desarrollo RUP en el flujo de trabajo de Pruebas.
- Identificar el proceso de Pruebas de Liberación en la Facultad 6.
- Selección de las pruebas a aplicar.
- Confección del Plan de Pruebas.
- Diseño de los casos de prueba.
    - Aplicación o realización de las pruebas escogidas.
- Elaboración del Informe de No Conformidades.
- Evaluación de los resultados obtenidos.

Este trabajo de diploma se encuentra estructurado en tres capítulos.

Capítulo 1: Fundamentación teórica. Se expone un basamento teórico y conceptual acerca de la situación real del aseguramiento de la calidad del software, las características fundamentales de las pruebas a aplicar para el éxito del sistema a liberar, incluyendo, las características fundamentales de la herramienta que se empleará para garantizar un exitoso despliegue del sistema informático que se vaya a implantar.

Capítulo 2: Diseño y aplicación de las pruebas. Se presentará el diseño de las pruebas que serán aplicadas al software “alasLIPO” partiendo del plan de prueba que se desarrolla, así mismo se realizará una descripción completa del sistema al cual se le aplicarán las pruebas.

Capítulo 3: Análisis de los resultados y valoración de la propuesta. En este capítulo se resumen las No Conformidades encontradas durante la aplicación de las pruebas y sobre los resultados de las pruebas de carga y estrés arrojados por la herramienta seleccionada, se realiza una valoración sobre la propuesta efectuada y se exponen las recomendaciones necesarias para la conclusión del trabajo realizado.

Capítulo1.
Fundamentación Teórica

En el presente capítulo se pretende sentar las bases que fundamentan el desarrollo de la tesis. Se parte de realizar una panorámica actual referente a la Calidad de Software, desde sus antecedentes hasta su definición, analizando los conceptos desarrollados por múltiples autores.

Además, se realiza un acercamiento al conocimiento de los Niveles y Técnicas, Métodos de Prueba, Pruebas de Rendimiento y Herramientas para su posterior ejecución, útiles para el desarrollo del proceso de prueba propuesto.

1.1 Calidad de Software. Panorámica actual en el mundo, Cuba y la UCI

En la actualidad una de las piezas más importantes en el engranaje de la economía mundial son las empresas de mediano y pequeño formato. En los últimos veinte años la industria del software ha surgido, desarrollado y consolidado a tal punto que hoy en día representa para la gran mayoría de los países del mundo una actividad económica de gran importancia, compuesta fundamentalmente por desarrolladoras de software que favorecen el crecimiento de las economías nacionales. *La mayoría de empresas desarrolladoras de software son pequeñas (tienen menos de 50 empleados) y desarrollan productos significativos que, para su construcción, necesitan prácticas eficientes de Ingeniería del Software adaptadas a su tamaño y tipo de negocio.* [1]

Argentina es uno de los países que ofrecen condiciones para convertirse en un productor de software en el mercado mundial, avalado en las disimiles investigaciones y numerosos resultados alcanzados. En este empeño juegan un papel fundamental las Universidades, por ejemplo *la Universidad Austral (UA) de argentina creó en octubre del 2006 un laboratorio de calidad del software con el propósito de realizar un aporte diferencial y ayudar así a que la industria nacional del software sea respetada en todo el mundo.* [2]

Desde sus inicios y hasta la actualidad la industria del software ha tenido que lidiar con graves problemas, entre los que figuran: las insatisfacciones de las expectativas de los clientes, la duración de los proyectos, siempre superiores a la estimación realizada inicialmente, así como el costo de estos,

también superiores a los estimados, siendo la causa de estos inconvenientes el hecho de que aun se considera el desarrollo del software como una tarea artesanal en vez de una ingeniería, de manera que toda industria para ser efectiva y eficiente necesita una ingeniería que la soporte y a su vez esta ingeniería trabaja con modelos de calidad, la clave del éxito no radica en crear nuevos modelos, que ya existen en abundancia, sino saber ponerlos en práctica.

Cuando se habla de realizar una buena práctica en la utilización de metodologías para medir la calidad del software, según estudios realizados, se puede decir que las empresas españolas encabezan los países europeos que siguen un buen modelo de calidad, a diferencia del resto de las del propio continente que aun tienen mucho camino por recorrer.

Más de la mitad de las empresas hoy, fallan a la hora de aplicar una metodología para asegurar la calidad del software que produce. Dentro de las principales causas de estos fracasos está el hecho de que las organizaciones permiten y aprueban que los equipos de desarrollo que tienen como objetivo trabajar en la calidad del software carezcan de formación y experiencia, pero sin embargo no emplean recursos para la capacitación de este personal, sin darse cuenta que el costo por los errores que cometen estos mismos equipos de desarrollo pueden ser mayores.

La informatización de la sociedad cubana, es una de las prioridades de nuestro gobierno revolucionario, con el objetivo de lograr esta meta en el país se está trabajando en función de aumentar el desarrollo de la Industria Cubana del Software, que permita no solo abastecernos de sistemas de información en beneficio de la sociedad cubana sino que podamos también exportar nuestros propios productos, logrando su reconocimiento internacional, sin embargo *las empresas que producen software hoy requieren aún de un sistema que cree un entorno organizado donde sean aplicados procedimientos de ingeniería de software que guíen el control de los procesos para desarrollar y mantener el software con una total calidad, para lograr evolucionar hacia una cultura de ingeniería de software y de administración de excelencia.* [3]

La Universidad de las Ciencias Informáticas (UCI) se encuentra enmarcada dentro de toda esta revolución informática en la que se halla inmerso nuestro país. Este centro fue creado bajo dos ideas claves: la informatización de la sociedad cubana y la exportación. Esferas como la Salud Pública se han beneficiados ya con los logros obtenidos, pudiendo plantear sin temor a equivocarnos que aun se están limando detalles en lo que a calidad de software se refiere.

En nuestra universidad se creó centralmente un grupo de trabajo especializado y en formación constante para centrar todo lo relacionado con la calidad de software. Subordinado a este grupo se crea uno de similar estructura en nuestra Facultad 6, con el objetivo de garantizar la calidad de los productos de software que se han y están desarrollando, apreciando de esta manera que un tema tan importante como la calidad no se ha dejado atrás. Todo esto se evidencia en los pequeños pero significativos aportes que se han logrado hacer por parte de los estudiantes y profesores que lo integran.

1.2 Pruebas de Software. Antecedentes. Actualidad en Cuba y la UCI.

El desarrollo de pruebas de software tiene sólo algunas décadas; la industria del software está aún definiendo sus caminos, buscando la manera adecuada de desarrollarse. Considerar las propuestas basadas en la experiencia de los demás permite un desarrollo con mayor velocidad.

Anteriormente, las pruebas de software se consideraban sólo una actividad que realizaba el programador para encontrar fallas en sus productos; con el paso de los años se ha determinado la importancia que tienen para garantizar el tiempo, el costo y la calidad del producto, de tal forma que actualmente son un proceso cuyo propósito principal es evaluar la funcionalidad del software respecto de los requerimientos establecidos al inicio.

La tendencia de las pruebas de software es iniciarlas antes, dentro del proyecto, y capacitar a especialistas responsables de esta actividad. El primer punto quiere decir que actualmente las especificaciones de pruebas se realizan al mismo tiempo que el diseño de software; la propuesta es iniciar el análisis del testware junto con el análisis del software. Es decir, sondeos preventivos que permitan ejecutar las pruebas tan pronto como el software esté listo y con ello no sólo descubrir errores, sino evitarlos. El segundo punto se refiere a crear conciencia acerca de la importancia de las pruebas y tener un equipo de personas dedicadas a esta actividad, que puedan integrarse a un proyecto y sean responsables de su calidad.

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

Por otro lado, se debe reconocer que las pruebas son una especie de administrador de riesgos: al igual que en los problemas de combinatorias complejas, se puede definir cuál debe considerarse buen resultado, aunque no necesariamente sea el mejor; es decir, que las pruebas sólo deben obtener un producto práctico con la calidad y funcionalidad requeridas. Cuando aparecieron los primeros grandes sistemas informáticos se incluyó a nivel metódico e imprescindible hasta entonces, un nuevo proceso en la confección de los mismos: el proceso de prueba.

Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del coste de un programa, ya que requiere un tiempo similar al de la programación, lo que obviamente acarrea un alto costo económico cuando estos no involucran vidas humanas, puesto que en este último caso el costo suele superar el 80%, siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema.

Un proceso de pruebas requiere mucho más que tiempo y dinero, necesita una verdadera metodología, la cual exige herramientas y conocimientos destinados a dicha tarea.

En Cuba hoy en día no existe una gran cultura sobre los temas de calidad, concretamente en lo que a Pruebas de Software se trata. Durante el desarrollo del software no existen en los grupos de desarrollo de los proyectos personas que desempeñen propiamente el rol de probador, esto trae consigo que en la etapa que corresponde probar el sistema no se cumpla con lo establecido para ello.

Las pruebas de caja blanca y caja negra generalmente no se le aplican a los sistemas, es válido aclarar que las pruebas de caja negra son realizadas con más frecuencia, aunque la documentación de ambas no se realiza con la calidad que es debida y en otros ni se hace.

Dentro de las causas principales del incorrecto cumplimiento de los criterios de desarrollo, con un mayor grado de importancia se encuentra la falta de una adecuada estructura organizativa dentro de las empresas para este efecto, la inexperiencia y el desconocimiento del tema o técnica a aplicar, el escaso personal capacitado y disponible para ello y el escaso tiempo del que se dispone para este proceso.

La calidad de software en Cuba es un tema que se encuentra aún en sus inicios, en la fase de estudio. Aunque muchas de las empresas ya están comenzando a profundizar en él, todavía falta mucho por

conocer en este sentido para lograr hacer de los sistemas nacionales, productos estables y con una calidad comercializable a cualquier nivel.

En la UCI, por su parte, los recursos humanos y materiales existentes facilitan el proceso de pruebas a los productos que se desarrollan. La estructura de calidad de software a nivel central y en cada una de las facultades permite la administración de calidad y la inserción desde etapas tempranas del desarrollo, de probadores a los distintos proyectos.

Se han alcanzado resultados que demuestran el avance progresivo en este tema, como por ejemplo, las pruebas realizadas a los diferentes proyectos en las distintas facultades, ejemplo de ello es el que ocupa este trabajo de diploma, que constituye un compromiso de la facultad con instituciones nacionales, para que estas también se vean beneficiadas.

Para la realización del proceso de prueba de liberación en la Facultad 6 se parte de una solicitud que envían los proyectos para liberar los productos y a partir de dicha solicitud se preparan las pruebas, se elabora un plan de pruebas que se discute y aprueba en una reunión de inicio.

En dicha reunión de inicio se establecen los protocolos de comunicación con el proyecto para la gestión de los entregables y de las no conformidades, para entonces de esta forma ejecutar las pruebas, se gestionan todas las no conformidades por iteraciones, lo óptimo es que aproximadamente se llegue hasta una tercera iteración, en función del estado del producto. Relacionado con este tema existen criterios de criticidad por los cuales un proceso de prueba puede abortar, o sea, se detiene la ejecución de las pruebas hasta tanto el proyecto no resuelva los problemas detectados.

Cuando se ejecutan todas las pruebas al 100% y se gestionaron y resolvieron todas las no conformidades, se decide entonces liberar el producto, emitiendo un acta de liberación, que constituye un aval para posteriores pruebas de aceptación o para la utilización efectiva del producto.

1.3 Calidad de Software. Conceptos y Definiciones.

Para nadie es un secreto que la calidad es sinónimo de eficiencia, flexibilidad, corrección, portabilidad, usabilidad, seguridad, integridad, etc. La calidad del software es medible y varía de un sistema a otro o de un programa a otro. Es posible medir los principales atributos que forman o caracterizan a un software de buena calidad. La idea es que la calidad del software se caracteriza por ciertos atributos: fiabilidad, flexibilidad, robustez, comprensión, adaptabilidad, modularidad, complejidad, portabilidad,

usabilidad, reutilización, eficiencia, y sin lugar a dudas es posible medir cada uno de ellos, y por consiguiente, caracterizar o medir la calidad del software en cuestión.

Para cada atributo a medir, hay una medición confiable (objetiva) que puede llevarse a cabo. La idea es que, dado que el atributo deseable es difícil de medir, se mide otro atributo que está correlacionado con el primero, por ejemplo:

- ✓ La fiabilidad (software confiable, pocos errores) se mide a través del número de mensajes de error, mientras más mensajes existan, contiene entonces menos errores este software.
- ✓ La flexibilidad (capacidad de adaptación a diferentes tipos de uso, a diferentes ambientes de uso) o adaptabilidad.
- ✓ La robustez (pocas fallas catastróficas, el sistema “no se cae”) se mide a través de pruebas y uso prolongado.
- ✓ La comprensión (capacidad de entender lo que el sistema hace) se mide viendo la cantidad de comentarios que posee el software, y la extensión de sus manuales de usuarios.
- ✓ El tamaño se mide en bytes, o sea el espacio que ocupa en memoria, (esta medición no tiene objeción, se mide lo que se quiere medir).
- ✓ La eficiencia de un programa se mide en segundos, es la rapidez en su ejecución (esta medición no tiene objeción, se mide lo que se quiere medir).
- ✓ La modularidad de un programa se mide contando el número de módulos que lo conforman.
- ✓ La complejidad de un programa se mide contando el número de anidaciones en expresiones o postulados, (es lo que se le llama complejidad ciclomática).
- ✓ La portabilidad es la facilidad con que se eche a andar en otro sistema operativo distinto al que fue creado. Se mide preguntando a usuarios que han hecho estos trabajos.
- ✓ La usabilidad de un programa es alta cuando el programa aporta gran valor agregado a nuestro trabajo. “Es indispensable contar con él”. Se mide viendo que porcentaje de nuestras necesidades cubre ese programa.
- ✓ La reutilización de un programa se mide por la cantidad de veces que partes del mismo se han reutilizado en otros proyectos de desarrollo de software.
- ✓ La facilidad de uso (ergonomía) caracteriza a los programas que no cuesta trabajo aprender, que se amoldan al modo intuitivo de hacer las cosas. Se mide observando la cantidad de pantallas que interaccionan con el usuario, y su sofisticación.

A lo largo de toda la historia, la búsqueda y el afán de perfección por parte del hombre ha sido constante, de tal forma, que el interés por el trabajo bien hecho y la necesidad de asumir

responsabilidades sobre la labor efectuada poco a poco derivó en el concepto de calidad [4], que con el tiempo ha adquirido un carácter multidimensional debido a que los diferentes autores, conocidos como los gurús del tema, lo han enfocado desde puntos de vistas diferentes: Edward Deming, lo enfoca como "*el grado predecible de uniformidad y conformidad a un bajo costo que se ajuste a las necesidades del mercado*"; Philip B. Crosby como *"cumplir con los requisitos";* y Joseph M. Juran como *"la idoneidad o aptitud para el uso"*.

Evidentemente todas esas definiciones son propias de autores que han investigado sobre el tema al igual que la planteada por Roger S. Pressman el cual expresó que la calidad de Software era *"la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente" [5];* esta definición es un poco mas abarcadora, porque la calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia, aunque la más utilizada es la brindada por la IEEE *"La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario".*[6].

El control de calidad de cualquier producto debe comprobar que la calidad del producto final se ajusta a una especificación dada. En el caso concreto del software, el control de calidad debe comprobar que el producto final funcione correctamente de acuerdo con sus especificaciones y en colaboración con otros sistemas software y bases de datos. [7]

Existen algunos criterios reconocidos sobre **Calidad**, por ejemplo:

- ✓ Adaptabilidad de uso" la cual implica dos parámetros fundamentales: calidad de diseño y calidad de conformidad, de manera que sea adaptable a los todos los usuarios. (Juran, 1970).
- ✓ Conformidad con los requisitos y la conformidad en el funcionamiento. (Deming).
- ✓ Cero defectos. (Crosby).
- ✓ Conformidad con los requerimientos. (Crosby, 1979).
- ✓ Pérdida económica que un producto supone para la sociedad desde el momento de su expedición. (Taguchi).
- ✓ Grado en el que un conjunto de características inherentes cumple con los requisitos. (ISO 9000-2000).
- ✓ Un buen producto no es el que cumple con una determinada especificación, sino el que es bien recibido por el cliente. (Ducker).

- ✓ La calidad del software está dada por la calidad de los procesos usados para desarrollarlo y mantenerlo. (Watts S. Humphrey).

Se puede tomar por conceptos de la Calidad los siguientes puntos:

- ✓ No es absoluta.
- ✓ Está sujeto a restricciones.
- ✓ Trata de compromisos aceptables.
- ✓ Es multidimensional.
- ✓ Los criterios de calidad no son independientes.
- ✓ La calidad está en permanente evolución.

Pero realmente ¿qué es la Calidad?, la pregunta que todos se hacen. Se puede decir entonces, según las definiciones reconocidas de algunas personalidades, que la **Calidad** es:

1. Conformidad con las especificaciones.
   - ✓ Control del proceso. (Shewart, Juran, Deming, Crosby).
   - ✓ Productos industriales. (Shewart, Juran, Deming, Crosby).
2. Satisfacción de las expectativas del cliente.
   - ✓ El mercado como eje de productos y servicios. (Davidow, Heshett).
   - ✓ Dificultad para medir las expectativas. (Davidow, Heshett).
3. Valor por dinero.
   - ✓ Mercado basado en precio y calidad. (Feigenbaum).
   - ✓ Preferencias del consumidor. (Feigenbaum).
4. Excelencia.
   - ✓ Aplicable en: productos, servicios, procesos y empresas. (Reeves, Bernard).
   - ✓ Lo mejor posible. (Reeves, Bernard).

Después del analizar qué es la Calidad, surge la incógnita: ¿existe diferencia entre **Calidad y Calidad Total**?

Para resolver esta incógnita se hace necesario conocer, qué es la **Calidad Total**, según Ishikawa, un autor reconocido de la gestión de la calidad, la define como: “Filosofía, cultura, estrategia o estilo de gerencia de una empresa según la cual todas las personas en la misma, estudian, practican, participan y fomentan la mejora continua de la calidad".

La calidad total es una alusión a la mejora continua, con el objetivo de lograr la calidad óptima en la totalidad de las áreas. Explica cómo ofrecer el mayor grado de satisfacción a un cliente por medio de un bien o servicio, medido por la conformidad del mismo. Siempre se está en constante perfeccionamiento manteniendo su objetivo, el cual debe ser alcanzable en la medida en que la necesidad de los clientes se satisface; una mayor satisfacción del cliente crea una mayor percepción de la calidad en el bien o servicio. En el momento que se satisface la necesidad de un cliente de forma total, se estará dando un producto de calidad total, entendiendo esto como el momento en que se satisface una necesidad anteriormente dada.

Después de estas importantes definiciones se hace alusión a la Calidad del Software como tal, de la misma se puede decir que no cuenta con una definición estándar, pues cada investigador la define desde su punto de vista, por esa causa no existe una específica; pero sí hay algunas que deben estar presentes en los procesos de desarrollo de software (ya sea distribuido o centralizado), sin dejar a un lado que cada institución use su propia versión, por ejemplo:

- ✓ “La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. (IEEE, Std. 610-1990).
- ✓ “Conjuntos de características de una entidad producto o servicio, que le confieren su amplitud para satisfacer necesidades expresadas e implícitas“. (ISO 8402- UNE 66-001-92).
- ✓ “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“. (Pressman, 1992).
- ✓ “Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario”. (Pressman, 1998).

Después de estas definiciones queda bien claro que la Calidad del Software es un proceso muy importante; y que alcanzarla se convierte en un gran objetivo que demanda mejoras en los resultados finales y por consiguiente un esfuerzo continuo. La calidad se evidencia en el empleo del software y se alcanza en cada contexto de uso reflejando el grado de satisfacción del cliente; para lograr que funcione como un todo, solo se hace necesario comprender las necesidades reales de los clientes con tantos detalles como sea posible.

1.4 El proceso de pruebas en RUP (Proceso Unificado de desarrollo). Aspectos generales, artefactos, trabajadores y actividades. Metodología.

RUP es una metodología muy potente de desarrollo de software, que describe como reutilizar los componentes existentes o implementar nuevos componentes con tareas bien definidas, el que nos lleva a obtener un sistema fácil de mantener, al que se le puede ir incrementando las posibilidades de reutilización. Asegurando la producción de software de alta calidad con un costo y tiempo predecible para el usuario.

Esta metodología divide el proceso de desarrollo en ciclos, donde se obtiene un producto al final de cada ciclo. Cada ciclo se divide en cuatro Fases:

- ✓ **Inicio:** El Objetivo en esta etapa es determinar la visión del proyecto.
- ✓ **Elaboración:** En esta etapa el objetivo es determinar la arquitectura óptima.
- ✓ **Construcción:** En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- ✓ **Transición:** El objetivo es llegar a obtener el release del proyecto.

Cada fase concluye con un hito bien definido donde deben tomarse ciertas decisiones.
En varios de los documentos estudiados se plantea la importancia que le da RUP a la ingeniería de pruebas, en aras de lograr la máxima calidad durante el desarrollo del producto en construcción, por ello define claramente aspectos como los objetivos de las pruebas, los principales artefactos y trabajadores, así como las actividades que deben desarrollar los ingenieros de pruebas en las 4 fases del ciclo de vida del proyecto.

RUP describe cómo planear y ejecutar las pruebas. Su puesta en práctica persigue varios objetivos que consideramos sean de vital importancia, por lo que se hace necesario:

- ✓ Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.

- ✓ Diseñar y crear las pruebas creando los casos de prueba que especifican qué probar, creando los procedimientos de prueba que especifican cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.

- ✓ Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente

devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados. [8]

1.4.1 Artefactos

Los principales artefactos definidos en RUP son:

1. Modelo de pruebas: Este modelo describe principalmente cómo se prueban los componentes ejecutables (como las construcciones) en el modelo de implementación con pruebas de integración y de sistema. Describe además cómo deben probarse aspectos específicos del sistema (interfaz de usuario, manual del usuario, etc.). El modelo de pruebas es un conjunto de casos de prueba, procedimientos de prueba y componentes de prueba.

2. Caso de prueba: Un caso de prueba específica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. Puede derivarse de un caso de uso del modelo de casos de uso o de una realización de caso de uso del modelo de diseño.

- ✓ Una **prueba de caja negra**, evalúa el comportamiento externamente observable del sistema, se puede realizar a partir de un caso de prueba que especifica cómo probar un caso de uso del modelo de casos de uso (verificación del resultado de la interacción entre los actores y el sistema, satisfacción de las precondiciones, poscondiciones y secuencias de acciones especificadas por el caso de uso).
- ✓ Una **prueba de caja blanca**, evalúa la interacción interna entre los componentes del sistema. Puede incluir la verificación de la interacción entre los componentes que implementan dicho caso de uso.

3. Procedimiento de prueba: Específica cómo realizar uno o varios casos de prueba o partes de éstos.

4. Componente de prueba: Automatiza uno o varios procedimientos de prueba o parte de ellos. Pueden ser programados o utilizar una herramienta de automatización de pruebas.

5. Plan de pruebas: Describe la estrategia, recursos y planificación de la prueba. La estrategia de prueba incluye la definición del tipo de prueba a realizar en cada iteración y sus objetivos.

6. Defecto: Es una anomalía del sistema, descubierto en una revisión o un fallo del software.

7. Evaluación de pruebas: Es una evaluación de los resultados de los esfuerzos de prueba. [8]

1.4.2 Trabajadores

Los trabajadores definidos en RUP son:

1. Diseñador de pruebas: Planifica las pruebas. Es responsable de la integridad del modelo de pruebas asegurando que el modelo cumple con su propósito. Del modelo de pruebas deben definir y describir los casos de prueba y los procedimientos de prueba. Además son los responsables de la evaluación de las pruebas de integración y de sistema cuando éstas se ejecuten.

2. Ingeniero de componentes: Son responsables de los componentes de pruebas que automatizan algunos de los procedimientos de pruebas.

3. Ingeniero de pruebas de integración: Son los responsables de realizar las pruebas de integración, las cuales se realizan para verificar que los componentes integrados en una construcción funcionan correctamente juntos, y se derivan a menudo de los casos de pruebas que especifican cómo probar realizaciones de caso de uso de diseño. Debe documentar los defectos en las pruebas de integración.

4. Ingeniero de pruebas de sistema: Son los responsables de realizar las pruebas de sistema sobre los ejecutables. Estas pruebas se llevan a cabo fundamentalmente para verificar las interacciones entre los actores y el sistema. Estas pruebas se derivan a menudo de los casos de pruebas que especifican cómo probar los casos de uso, también se usan para probar el sistema como un todo. Debe documentar los defectos en las pruebas de sistema. [8]

1.4.3 Actividades.

Las principales actividades definidas en RUP se explican a continuación:

1. Planificar pruebas

Al planificar las pruebas, se planifican los esfuerzos de prueba en una iteración y debe llevarse a cabo las siguientes tareas:

- ✓ Describir una estrategia de prueba: cuantos flujos básicos y alternativos deben ser probados, cuantas pruebas se realizarán automatizadas y cuántas manuales.
- ✓ Estimar los requisitos para el esfuerzo de la prueba: recursos humanos, sistemas necesarios, etc.
- ✓ Planificar el esfuerzo de prueba.

Para obtener el plan de prueba, la estrategia de pruebas y los requisitos necesarios de pruebas de una iteración los ingenieros de prueba deben partir de los artefactos de entrada: requisitos adicionales, modelo de casos de uso, modelo de análisis, modelo de diseño, modelo de implementación, descripción de la arquitectura (vistas arquitectónicas de los modelos).

2. Diseñar pruebas

Al diseñar las pruebas deben tenerse en cuanta los siguientes propósitos:

- ✓ **Identificar y describir los casos de prueba para cada construcción**

Para diseñar las pruebas de cada construcción los ingenieros de prueba deben partir de los artefactos de entrada: requisitos adicionales, modelo de casos de uso, modelo de análisis, modelo de diseño, modelo de implementación, descripción de la arquitectura (vistas arquitectónicas de los modelos) y del plan de pruebas (estrategia y planificación). De esa forma deben obtener los casos de pruebas y los procedimientos de pruebas.

- ✓ **Diseño de casos de pruebas de integración.**

Deben ser diseñados los casos de prueba de integración para verificar que los componentes interaccionan entre sí de la forma apropiada después de ser integrados en una construcción.

Los diseñadores de pruebas buscan combinaciones de entradas, salida y estado inicial de sistema que den lugar a escenarios interesantes que empleen las clases (y por tanto los componentes) que participan en los diagramas.

- ✓ **Diseño de casos de prueba de regresión.**

Algunos de los casos de pruebas de construcciones anteriores pueden ser usados para pruebas de regresión en construcciones subsiguientes, aunque no todos pueden ser utilizados para pruebas de regresión. Los casos de pruebas, para ser usados en pruebas de regresión, deben ser suficientemente flexibles para ser resistentes a cambios, esta flexibilidad debe ser cuidadosa ya que la conversión de un caso de prueba a caso de prueba de regresión supone un esfuerzo de desarrollo extra, luego se debe convertir casos de prueba a casos de prueba de regresión sólo cuando el esfuerzo merezca la pena.

Se debe reutilizar procedimientos de prueba existentes tanto como sea posible, realizando las modificaciones adecuadas. Los diseñadores de prueba deben crear procedimientos de pruebas que puedan ser reutilizados en varios casos de prueba.

3. Implementar pruebas

El propósito de la implementación de pruebas es automatizar los procedimientos de prueba, creando componentes de pruebas cuando sea posible.

4. Realizar pruebas de integración

Esta actividad garantiza que se realicen las pruebas de integración necesarias para cada una de las construcciones creadas en una iteración y se recopilan los resultados de las pruebas. Estas pruebas se llevan a cabo con los siguientes pasos:

- ✓ Realizar las pruebas de integración realizando los procedimientos de pruebas manualmente o ejecutando los componentes de pruebas si existen.
- ✓ Comparar los resultados de las pruebas con los esperados e investigar los resultados de las pruebas que no coinciden con los esperados.
- ✓ Informar de los defectos a los ingenieros de componentes
- ✓ Informar de los defectos a los diseñadores de pruebas, quienes usarán los defectos para evaluar los resultados del esfuerzo de prueba.

5. Realizar prueba de sistema

La prueba de sistema puede empezar cuando las pruebas de integración indican que el sistema satisface los objetivos de calidad de integración fijados en el plan de pruebas de la iteración. Debe realizarse con pasos análogos a las pruebas de integración.

6. Evaluar prueba

El propósito de esta actividad es evaluar los esfuerzos de prueba en cada iteración. Para evaluar los resultados de pruebas estos deben ser comparados con los objetivos esbozados en el plan de pruebas. Los diseñadores de pruebas preparan métricas que les permiten determinar el nivel de calidad de software y qué cantidad de pruebas es necesario hacer. [8]

Luego de analizar el proceso de pruebas que define RUP, se decide utilizar para la ejecución de las pruebas el proceso que describe RUP. Esta metodología se puede aplicar en proyectos grandes y pequeños, muy potente para el desarrollo de software basada en UML (Lenguaje Unificado de Modelado), que describe como reutilizar los componentes existentes o implementar nuevos componentes existentes con tareas bien definidas, llevándonos a obtener un sistema fácil de mantener al que se le puede ir incrementando las posibilidades de reutilización.

Una de las grandes ventajas que tiene RUP son sus casos de uso y sus casos de prueba, ya que los casos de uso facilitan las tareas de programación y los casos de prueba garantizan un plan de pruebas bastante bueno y robusto. Otra ventaja y que constituye una de sus características principales es su enfoque iterativo, implicando con esto que se estará evaluando a lo largo de todo el proyecto, con el fin de encontrar defectos lo antes posible, y poder así reducir el costo por la detección de defectos.

1.5 Niveles de Prueba

En el proceso de pruebas al software existen una serie de niveles en los cuales se ejecutan diferentes tipos de pruebas de acuerdo al desarrollo que se haya alcanzado en la aplicación. Estos niveles especifican qué tipos y métodos de pruebas se deben utilizar en cada uno de ellos y cuáles son sus objetivos. Los niveles de prueba por los que transita un software son:

- ✓ De Desarrollador
  - ➢ Diseñada e implementada por el equipo de desarrollo.
  - ➢ Tradicionalmente consideradas solo para la prueba de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad. [9]

- ✓ De Independiente
  - ➢ Diseñada e implementada por alguien independiente del grupo de desarrollo.
  - ➢ Su objetivo es proporcionar una perspectiva diferente y en un ambiente más rico que el de los desarrolladores. Una vista de la prueba independiente es la prueba independiente de los stakeholders, que son pruebas basadas en las necesidades y preocupaciones de los stakeholders. [9]

- ✓ De Unidad
  - ➢ Enfocadas al código fuente de los componentes.
  - ➢ Para verificar todos los flujos de control.
  - ➢ Primero pasa por la revisión del programador. [9]

- ✓ De integración
  - ➢ Prueba los componentes combinados para ejecutar una funcionalidad.
  - ➢ Para verificar descubrir errores o incompletitud en las especificaciones de las interfaces de las clases. [9]

- ✓ De sistema
    - ➢ Prueba el software funcionando como un todo.
    - ➢ Aceptable para cuando el software se encuentra en la Fase de Construcción.
    - ➢ Trata de probar que los objetivos para los que fue construida la aplicación no se cumplen en su totalidad y que por tanto hay que cambiar cosas en la aplicación.
    - ➢ Se usa como base los objetivos originales.
    - ➢ No existe un método en específico, sino que se dan lineamientos, a la hora de preparar los casos de prueba.
    - ➢ Se finaliza cuando se cumple el tiempo estimado y se han detectado N errores. [9]

- ✓ De aceptación
    - ➢ Prueba el software funcionando como un todo, se realiza dándole un uso real a la aplicación y es realizada por parte de los clientes, los errores que se encuentren en la misma son reportados como defectos. [9]

Las pruebas del software se encuentran en cada etapa del desarrollo del software. En la medida que el trabajo de los desarrolladores va aumentando el volumen de la aplicación, las pruebas van cambiando de estrategias y técnicas, presentándose como una nueva etapa, estas están definidas en niveles que tienen nuevos objetivos, entornos y resultados.

En el caso del sistema alasLIPO las pruebas se desarrollaron a nivel de Sistema.

1.6 Técnicas de Prueba

Bajo el nombre de pruebas de software se agrupan un conjunto de prácticas correctivas, frente a las prácticas preventivas que se aplican durante el proceso de construcción de software, cuyo objetivo es determinar la calidad de los sistemas de software, existen diferentes técnicas de pruebas de software y se pueden agrupar en las siguientes categorías:

- ✓ Técnicas Aleatorias: Los casos que se prueban se generan aleatoriamente.
- ✓ Técnicas Funcionales: Se utilizan las especificaciones del problema para generar los casos de prueba. El programa se ve como una caja negra.
- ✓ Técnicas de Flujo de Control: Se requiere conocimiento del código fuente. Se seleccionan caminos dentro del programa que deben ser ejecutados al ingresar los casos de prueba.

- ✓ Técnicas de Flujo de Datos: También requiere conocimiento del código fuente. En este caso, los caminos se eligen en forma de explorar secuencias de eventos relacionadas con el estado de las variables.
- ✓ Técnicas de Mutación: En la mutación se introducen fallas al programa creando varios mutantes, cada uno con una falla. Los casos de prueba se hacen pasar por los mutantes con la intención de hacer fallar al programa.

1.7 Métodos de Prueba

Dentro de los métodos más sobresalientes a la hora de realizar pruebas se encuentran el método de Caja Blanca y el método de Caja Negra.

Cualquier producto ingenieril puede ser probado de dos formas:

1. Conociendo el funcionamiento interno de un producto, las pruebas pueden ser llevadas a cabo para asegurar que todas las piezas encajen; esto es, que la operación interna del producto se lleve a cabo de acuerdo a las especificaciones y que todos los componentes internos hayan sido ejecutados adecuadamente; a esto se le llama pruebas de Caja Blanca.
2. Conociendo la función específica que el producto debe de realizar, las pruebas pueden ser llevadas a cabo para demostrar que cada función es completamente operacional; a esto se le llama pruebas de Caja Negra.

Como método de prueba escogido para realizar la liberación en alasLIPO se definió el Método de Prueba de Caja Negra.

1.7.1 Prueba de Caja Blanca

Se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

En estas pruebas estamos siempre observando el código, que las pruebas se dedican a ejecutar con ánimo de "probarlo todo". [10]

Cuando se implementa un software se pretende garantizar la obtención de un producto con calidad, para lo que resulta recomendable comprobar que el código que se ha escrito funciona correctamente. Con este propósito se implementan pruebas que verifican que el programa genera los resultados que realmente esperamos. Las pruebas realizadas al código son las llamadas pruebas de caja blanca, y se basan en un examen minucioso de los detalles procedimentales, logrando examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. Estas no se deben confundir con las pruebas informales que realiza el programador mientras está desarrollando el módulo.

Las pruebas de caja blanca han tomado un lugar muy importante en el desarrollo de sistemas de cualquier tipo, estas pruebas constituyen un método mediante el cual el ingeniero del software puede obtener casos de prueba que garanticen:

- ✓ Ejercitar por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Ejercitar todas las decisiones lógicas en sus vertientes verdadera y falsa.
- ✓ Comprobar los ciclos en sus límites y con sus límites operacionales.
- ✓ Ejercitar las estructuras de datos internas para asegurar su validez [11].

El principal factor que se debe considerar al inicio de las pruebas es el tamaño del módulo a probar en caso de que el proyecto esté dividido en diferentes módulos, se debe considerar si el tamaño del módulo permitirá probar adecuadamente toda su funcionalidad de manera sencilla y rápida.

También es importante separar los módulos de acuerdo a su funcionalidad, si los módulos son muy grandes y contienen muchas funcionalidades, estos se volverán más complejos de probar y al encontrar algún error será más difícil ubicar la funcionalidad defectuosa y corregirla. Al hacer esta labor el analista de pruebas podrá recomendar que un módulo muy complejo sea separado en 2 o 3 módulos más sencillos.

Cada método de una clase puede ser sometido a pruebas para determinar si está correctamente implementado de acuerdo con la especificación que contiene el modelo funcional. Algunos métodos son sencillos, los cuales no conllevan la realización de muchas pruebas.

En la Figura # 1.1 se muestra lo que se considera una representación clásica de pruebas de caja blanca. En este tipo de pruebas el cubo representaría un sistema en donde se pueden observar los diversos componentes que forman parte del mismo, cada uno de estos componentes debe ser probado

en su totalidad (óvalos), también sus interfaces o comunicaciones con los demás componentes (flechas), este tipo de pruebas también son llamadas pruebas de caja de cristal ya que este último término representa mejor el tipo de pruebas.

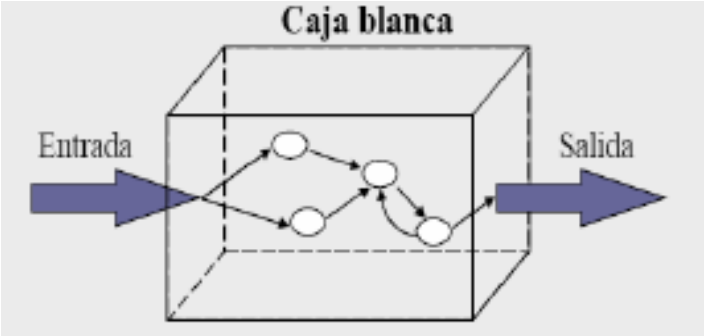

Fig. 1.1 Prueba de Caja Blanca

Cuando se aplican estas pruebas se debe tener en cuenta los siguientes aspectos:

- ✓ Los datos de entrada son conocidos por el Probador o Analista de Pruebas y estos deben ser preparados con minuciosidad, ya que el resultado de las pruebas depende de estos.
- ✓ Se debe conocer que componentes interactúan en cada caso de prueba.
- ✓ Se debe conocer de antemano que resultados debe devolver el componente según los datos de entrada utilizados en la prueba.
- ✓ Finalmente se deben comparar los datos obtenidos en la prueba con los datos esperados, si son idénticos podemos decir que el módulo superó la prueba y empezamos con la siguiente.

Para lograr un buen resultado a partir de la aplicación de las pruebas de caja blanca, no se debe esperar a que esté todo el código escrito para comenzar a probar. Estas pruebas deben irse realizando según se va generando el código, facilitando con esto que los errores se detecten lo antes posible. Es recomendable que estas pruebas sean diseñadas y aplicadas por una persona distinta a la que escribe el código; siendo esta la única forma de no ser "comprensivo con los fallos".

Para aplicar las pruebas de caja blanca existen diferentes técnicas entre las que se encuentran la Técnica del Camino Básico, de la Condición, la de Bucles y la de Flujo de Datos.

1.7.2 Pruebas de Caja Negra

Las pruebas de caja negra (Fig. 8) se centran en los requisitos funcionales, permitiendo al ingeniero del software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Estas pruebas se llevan a cabo sobre la interfaz del software, y

es completamente indiferente al comportamiento interno y la estructura del programa. Los casos de prueba de caja negra pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta, y que se mantiene la integridad de la información externa.

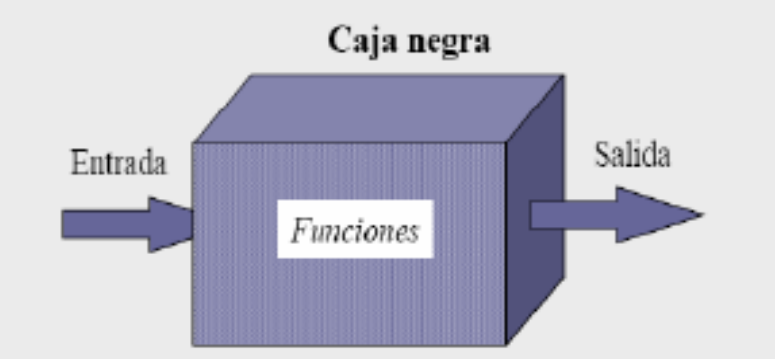

Fig. 1.2 Prueba de Caja Negra.

Estas pruebas tienen también como meta determinar la eficiencia del programa desde el desempeño en el equipo, el tiempo de retardo de las salidas hasta el nivel de recuperación del sistema luego de fallas o caídas, sean estas producidas por manejo incorrecto de datos, equipo, o producidas externamente.

- ✓ Los casos de prueba de caja negra pretende demostrar que:
- ✓ Las funciones del software son operativas.
- ✓ La entrada se acepta de forma adecuada.
- ✓ Se produce una salida correcta, y
- ✓ La integridad de la información externa se mantiene [12].

La prueba de caja negra intenta encontrar errores de los siguientes tipos:

- ✓ Funciones incorrectas o inexistentes.
- ✓ Errores relativos a las interfaces.
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas.
- ✓ Errores debidos al rendimiento.
- ✓ Errores de inicialización o terminación.

Existen diferentes métodos que se utilizan en la aplicación de las pruebas de caja negra. El empleo de estos permite dado una serie de valores de entrada, obtener un conjunto de valores de salida y

además posibilita realizar una comparación entre varias versiones con los mismos datos de entrada, para poder verificar que las salidas sean las mismas. A continuación se hace un breve análisis de estos métodos.

1.7.2.1 Técnica de Caja Negra.

1.7.2.1.1 Partición Equivalente.

Pressman presenta la partición equivalente dentro del Método de Prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases.

La partición equivalente es subjetiva. Dos probadores quienes prueban un programa complejo pueden llegar a diferentes conjuntos de particiones.

En el diseño de casos de prueba para partición equivalente se procede en dos pasos:

1 Se identifican las clases de equivalencia. Las clases de equivalencia son identificadas tomando cada condición de entrada (generalmente una oración o una frase en la especificación) y repartiéndola en dos o más grupos.

   Es de notar que dos tipos de clases de equivalencia están identificados: las clases de equivalencia válidas representan entradas válidas al programa, y las clases de equivalencia inválidas que representan el resto de los estados posibles de la condición (es decir, valores erróneos de la entrada).

2 Se define los casos de prueba. El segundo paso es el uso de las clases de equivalencia para identificar los casos de prueba. El proceso es como sigue: se asigna un número único a cada clase

de equivalencia. Hasta que todas las clases de equivalencia válidas han sido cubiertas por los casos de prueba, se escribe un nuevo caso de prueba que cubra la clase de equivalencia válida. Y por último hasta que los casos de prueba hayan cubierto todas las clases de equivalencia inválidas, se escribe un caso de la prueba que cubra una, y solamente una, de las clases de equivalencia inválidas descubiertas.

1.7.2.1.2 Análisis de Valores Límite.

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites (AVL) como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite.

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

Condiciones sublímate. Las condiciones limite normales son las más obvias de descubrir. Estas son definidas en la especificación o son evidentes al momento de utilizar el software. Algunos límites, sin embargo, son internos al software, no son necesariamente aparentes al usuario final pero aún así deben ser probadas por el probador. Estas son conocidas como condiciones sublímites o condiciones límite internas. Una condición sublímites común es la tabla de caracteres ASCII, por ejemplo, si se está evaluando una caja de texto que acepta solamente los caracteres AZ y az, se debe incluir los valores en la partición inválida justo «debajo de» y «encima de» esos caracteres de la tabla ASCII.
Esta técnica de prueba complementa a la partición equivalente. En lugar de centrarse solamente en las condiciones de entrada, este obtiene también casos de prueba para el campo de salida.

Dentro de las reglas para el análisis de valores limites se encuentran:

- ✓ Si una condición de entrada específica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, y para los valores justo por debajo y justo por encima de a y b, respectivamente.
- ✓ Si una condición de entrada específica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo, uno más el máximo y uno menos el mínimo.
- ✓ Aplicar las directrices anteriores a las condiciones de salida.

- ✓ Si las estructuras de datos internas tienen límites preestablecidos hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.[13]

1.8 Pruebas de Carga y Estrés.

Prueba de Carga

Este tipo de prueba se enfoca en validar y evaluar aceptabilidad de un elemento de un sistema sobre diferentes cargas de trabajo mientras el sistema permanece constante. Generalmente se incluye simulación de cargas de trabajo promedio y pico que puedan ocurrir dentro de la tolerancia operacional normal.

Pruebas de Estrés

Este tipo de prueba se enfoca a evaluar el comportamiento del sistema basado condiciones anormales. Stress del sistema se refiere a extrema carga, memoria insuficiente, no disponibilidad de servicios y hardware o recursos compartidos limitados. Este tipo de prueba permite comprender mejor cómo y qué áreas del sistema colapsarán, de este modo es posible planificar contingencias y actualizar el mantenimiento y planear y asignar recursos de antemano.

1.9 Herramientas usadas para realizar pruebas de software.

Dentro de las herramientas que se usan a nivel mundial para la realización de pruebas de software están:

- ✓ **JWebUnit:** Entorno de pruebas para Java creado por Erich Gamma y Kent Beck.
- ✓ **TestNG:** Creado para suplir algunas deficiencias en JWebUnit.
- ✓ **JTiger:** Basado en anotaciones, como TestNG.
- ✓ **SimpleTest:** Entorno de pruebas para aplicaciones realizadas en PHP.
- ✓ **Rational TestManager**. Consola central para la administración de la actividad, ejecución y reportes de test.
- ✓ **NUnit:** Migración del entorno JWebUnit para lenguajes de la plataforma .NET.
- ✓ **QALoad, LoadRunner**, **Quality Center**.
- ✓ **JMeter:** Herramienta Java desarrollada dentro del proyecto Jakarta para la realización de pruebas de carga y estrés.

1.9.1 JWebUnit

A lo largo de los años, han existido programadores experimentados que han desarrollado métodos y herramientas para escribir las pruebas más cómodamente.

Dos de estos desarrolladores fueron Kent Beck y Eric Gamma (dos personalidades en el campo de la informática, el primero por desarrollar la metodología Extreme Programming y el otro por su contribución al libro Design Patterns), quienes desarrollaron una colección de clases para Java llamada JUnit.

Con estas clases, se pueden desarrollar casos de pruebas y colecciones de prueba fácilmente, heredando de sus propias clases base. Además esta herramienta ofrece una serie de interfaces gráficas para visualizar estas pruebas, ejecutarlas, ver sus resultados, seleccionar aquellas que queremos ejecutar, etc.

Permite además realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

1.9.2 TestNG

TestNG es un marco de prueba inspirado de JUnit y de NUnit, pero se pueden introducir algunas nuevas funcionalidades que lo hagan más de gran alcance y más fácil utilizar, por ejemplo:
Anotaciones del JDK 5 (el JDK 1.4 también se apoya con las anotaciones de JavaDoc).

Configuración flexible de la prueba

- ✓ Ayuda para la prueba data-driven.
- ✓ Ayuda para los parámetros.
- ✓ Permite la distribución de pruebas en las máquinas auxiliares.
- ✓ Es un modelo de gran alcance de ejecución.
- ✓ Apoyado por una variedad de herramientas y de plug-ins (eclipse, IDEA, Maven, etc.).
- ✓ Encaja BeanShell para la flexibilidad adicional.
- ✓ Omite las funciones del JDK para el tiempo de pasada y registrar (ningunas dependencias).
- ✓ Métodos dependientes para la prueba del servidor de uso.

TestNG se diseña para cubrir categorías de pruebas como: unidad, funcional, integración, entre otras.

1.9.3 JTiger

El framework JTiger para el estándar Java 2 edición 1.5, es una solución del open source que proporciona una abstracción robusta y una buena característica para desarrollar y ejecutar casos de prueba de unidad. Los casos de prueba metadatos y la documentación de prueba de unidad se expresa con anotaciones.

1.9.4 SimpleTest

En el contexto del desarrollo ágil, el código de prueba está junto al código fuente y ambos se escriben simultáneamente. En este contexto SimpleTest apunta ser una solución completa de prueba del desarrollador PHP y se llama “simple” porque debe ser fácil de utilizar y de extender.
Incluye todas las funciones típicas de JUnit y de los puertos de PHPUnit, pero también agrega objetos falsos. También tiene cierta funcionalidad de JWebUnit.
Esta herramienta es utilizada para crear plan de pruebas y casos de pruebas.

1.9.5 Rational TestManager

A continuación se proporciona información sobre la herramienta Rational TestManager para la creación de un plan de prueba:

- ✓ Define el acercamiento de la prueba
- ✓ Identifica los motivadores de prueba.
- ✓ Identifica los blancos de prueba.

Rational TestManager es la consola central para la administración, ejecución y divulgación de la prueba. Construido para la extensibilidad, apoya los acercamientos de la prueba manual a los varios paradigmas automatizados incluyendo la regresión de prueba de unidad y funcional.

1.9.6 NUnit

NUnit es una de esas herramientas utilizada para escribir y ejecutar pruebas en .NET, es un framework desarrollado en C# que ofrece las funcionalidades necesarias para implementar pruebas en un proyecto. Además provee una interfaz gráfica para ejecutar y administrar las mismas. También se utiliza para realizar pruebas unitarias a .NET.

Lo que hace tan efectiva la TDD (Test Driven Development) es la automatización de las pruebas de programador, (programmer unit tests).

El hecho de utilizar TDD implica 3 acciones: escribir las pruebas, escribir el código que debe pasar las pruebas y refactorizar para eliminar el código duplicado. La primera se lleva a cabo de una manera sencilla y simple utilizando NUnit, esta soporta los lenguajes bases de .NET como C#, J#, VB y C++.

NUnit es una herramienta que se encarga de analizar ensamblados generados por .NET, interpretar las pruebas inmersas en ellos y ejecutarlas. Utiliza atributos personalizados para interpretar las pruebas y provee además métodos para implementarlas. En general, NUnit compara valores esperados y valores generados, si estos son diferentes la prueba no pasa, caso contrario la prueba es exitosa.

NUnit ofrece una interfaz simple que informa si una prueba falló, pasó o fue ignorada. Basa su funcionamiento en dos aspectos, el primero es la utilización de atributos personalizados. Estos atributos le indican al framework de NUnit que debe hacer con determinado método o clase, es decir, estos atributos le indican a NUnit como interpretar y ejecutar las pruebas implementadas en el método o clase. El segundo son las aserciones, que no son más que métodos del framework de NUnit utilizados para comprobar y comparar valores.

1.9.7 QALoad

QALoad de la empresa Compuware es una herramienta de automatización de pruebas de carga para web, Java, .NET, aplicaciones ERP (Planificación de recursos empresariales) y CRM (gestión de relaciones con los clientes y ambientes distribuidos). Simula miles de usuarios desempeñando transacciones de negocio clave de una aplicación para asegurar su desempeño y escalabilidad previa a su puesta en producción. Con esta herramienta, los equipos de prueba pueden rápidamente detectar problemas, optimizar el desempeño de los sistemas y ayudar a asegurar un despliegue de aplicaciones exitoso [19].

Compuware (NASDAQ: CPWR) es líder mundial en proporcionar software y servicios profesionales que permiten a las empresas gestionar sus negocios y maximizar el valor de sus activos de TI. Las soluciones de Compuware aceleran el desarrollo, aumentan la calidad y mejoran el rendimiento de los sistemas críticos, mejorando su eficiencia, el control de costos y la productividad de sus empleados a lo largo de toda la organización. Fundada en 1973, Compuware trabaja para las mayores organizaciones de TI del mundo, incluyendo a más del 90 por ciento de las empresas del Fortune 100[20].

1.9.8 LoadRunner

LoadRunner es una herramienta para realizar pruebas de carga de Mercury Interactive que permite pre-ver el comportamiento y el rendimiento del sistema. Además, permite poner a prueba toda la infraestructura corporativa para identificar y aislar los posibles problemas mediante la simulación de la actividad de miles de usuarios. Realiza pruebas en toda la infraestructura corporativa, que comprende las soluciones e-business, ERP, CRM y las aplicaciones personalizadas, simulando la actividad de miles de usuarios, con lo que los equipos de desarrollo de aplicaciones y sitios Web pueden mejorar el rendimiento de las aplicaciones. Es la herramienta de pruebas de carga más escalable que permite simular la actividad de miles de usuarios con los mínimos recursos de hardware. Se integra a la perfección con las herramientas de gestión del rendimiento de Mercury Interactive. Los mismos scripts creados durante las pruebas pueden volverse a utilizar para monitorizar la aplicación una vez terminada su implantación. Presenta una interfaz API abierta, con la que los usuarios y otros fabricantes pueden integrar LoadRunner en sus propios entornos [21].

Mercury Interactive empresa que realizó la herramienta Load Runner es una compañía especializada en Business Technology Optimization (BTO), española que recientemente ha sido comprada por HP [22].

1.9.9 Quality Center

Permite automatizar los procesos de calidad, uniendo todos los componentes con las aplicaciones correctas para acelerar los tiempos de depuración. El resultado es una mejora impresionante en la calidad y en la consistencia de la aplicación. Ayuda a manejar y controlar el riesgo, mientras se desarrolla y prueba la aplicación. En todos los momentos del proceso, se tiene visibilidad de en dónde se encuentra el proyecto con respecto a calidad (requerimientos probados y satisfechos, pruebas ejecutadas, defectos y tendencias encontradas, etc.). Maneja y automatiza el proceso de entrega, con indicadores claves de rendimiento, a tiempo real, proporciona aplicaciones que automatizan todas las actividades claves en los procesos de calidad, y apoya a todas las personas y roles que necesitan involucrarse en los procesos de entrega [23]. Esta herramienta no es una herramienta de pruebas de sistemas pero se ha decidido involucrarla en la investigación ya que la utilización de la misma en el manejo de la calidad del proceso de pruebas influye en la garantía de los resultados y del proceso en general, permitiendo una mayor organización del proceso e influye directamente en el rendimiento del equipo de trabajo.

1.9.10 JMeter

Es una herramienta Java desarrollada dentro del proyecto Jakarta, que permite realizar Pruebas de Rendimiento y Pruebas Funcionales sobre Aplicaciones Web [14]. JMeter permite realizar pruebas Web clásicas así como test.
No se necesita la utilización de una licencia ya que se permite bajar de manera gratuita el software desde la Web que representa al proyecto Jakarta.

Se puede decir además que existe una amplia documentación sobre la utilización el modo de empleo de la herramienta y los requerimientos no funcionales que ella precisa. Así mismo es válido referir que esta herramienta fue realizada en el marco del software libre.

1.10 JMeter. Herramienta Seleccionada para Automatizar Pruebas de Carga y Estrés.

Se ha decido utilizar para las pruebas Carga y Estrés la herramienta JMeter por las funcionalidades que presenta, además de ser un software libre y gratis.

La utilización del JMeter supone un 95 % de tiempo menos para la realización de estas pruebas. La mayor inversión de tiempo que se necesita para la realización de las pruebas es la fase de estudio de los casos de uso críticos en la aplicación y la elaboración del plan de pruebas en la herramienta. Permite almacenar los resultados de la prueba y generar gráficos que representan los aspectos que se han probado.

La posibilidad de contar con material para el estudio de la utilización de la herramienta, supone una garantía en el éxito de la ejecución de las pruebas. Otro aspecto a favor de esta herramienta es su desarrollo en el marco del software libre, política que apoya la universidad y que por las características existe un por ciento superior de confianza en la utilización de la misma ya que se puede contar con todo el código que genera a la herramienta.

Como desventaja de la herramienta se encuentra el alto consumo de rendimiento del CPU ya que es necesario para su mejor desempeño más de 512mb de RAM.

Dentro de las funcionalidades que brinda la herramienta JMeter se encuentran:

- ✓ Permite realizar test de FTP, JDBC, JNDI, LDAP, SOAP/XML-RPC, y WebServices (en Beta).
- ✓ Ofrece la posibilidad que el propio usuario desarrolle en Java un “Controller” a medida, cumpliendo una interfaz Java, y depositando el .jar correspondiente al desarrollo en el directorio “lib” de JMeter lo que permite reducir el consumo de rendimiento del CPU.

- ✓ Permite realizar pruebas distribuidas en distintos ordenadores que actuarán como clientes simulando varios hilos que harán función de usuarios.
- ✓ Permite generar un caso de prueba es a través de una navegación de usuario. [14].
- ✓ Tipos de informes que puede generar:
  - ❖ Assertion Results: Muestra la URL de cada petición e indica los errores que se produzcan (Assertions que no se han cumplido) en el test.
  - ❖ Graph Full Results: Simplemente muestra el tiempo.
  - ❖ Graph Results: Muestra un gráfico con los tiempos medio, desviación, throughput, etc. de la ejecución del plan de prueba.
  - ❖ Mailer Visualizar: Permite enviar un e-mail si el plan de pruebas falla o no, o supera un determinado valor de fallos o éxitos.
  - ❖ Simple Data Writer: Vuelca los resultados a un fichero.
  - ❖ Spline Visualizer: Gráfico de tiempos como spline.
  - ❖ Aggregate Report: Muestra una tabla con una fila por cada URL solicitada, indicando el tiempo min, máx, medio, etc. Es una tabla que totaliza por URL.
  - ❖ View Results in Table: Muestra una tabla con todas las respuestas, la URL, tiempo y resultado de ejecución de cada una de ellas.
  - ❖ View Results in Tree: Muestra un árbol con todas las respuestas y sus tiempos.
- ✓ Permite controlar todos los parámetros de una request http por ejemplo:
  - ❖ Método: GET o POST.
  - ❖ Path del recurso a pedir.
  - ❖ Redirección automática: Seguir las redirecciones indicadas por el resultado de la petición.
  - ❖ Use KeepAlive: Mantener la conexión viva entre distintas peticiones
- ✓ Permite envío de parámetros en la request.
- ✓ Permite envío de un fichero adjunto a la request.
- ✓ Permite especificar el número de threads (hilos de ejecución) en paralelo, así como el tiempo de arranque de cada uno, y número de iteraciones que hará cada uno de ellos (puede marcarse como infinito).
- ✓ Se puede planificar (scheduler, función automática para el arranque de la aplicación) la ejecución de la prueba indicando la hora de arranque y parada, o la duración del test en segundos y el tiempo de arranque del mismo.
- ✓ Permite la reutilización de los scripts realizados para la prueba.

1.11 Gestión de Calidad: Una vista desde el enfoque de la Guía del PMBOK.

La gestión de calidad de software cobra cada vez mayor importancia en las empresas que producen software por el impacto que tienen los errores identificados en cliente cuando no se llevan los procesos adecuados de calidad.

La gestión de la calidad del producto debe iniciarse desde las etapas tempranas del proyecto con el fin de que la actividad tenga mayor orientación a la prevención que a la corrección. Corregir los problemas en las etapas tempranas es más barato que cuando se identifican en etapas más avanzadas del proceso. El proceso de pruebas es parte de las actividades de Aseguramiento de la Calidad, pero no son suficientes para garantizar la gestión de calidad del proceso de desarrollo ni mucho menos del proyecto, por lo que se requieren complementar con actividades de calidad orientada a los procesos como ISO 9000.

A su vez la gestión de calidad de un proyecto de software al ser manejada como proyecto puede abordarse con el enfoque de la Guía del PMBOK, involucrando las áreas de conocimientos que propone dicha guía.

La Guía de PMBOK es un estándar en la gestión de proyectos desarrollado por el PMI (Project Management Institute). En un intento por documentar y estandarizar información y prácticas generalmente aceptadas en la gestión de proyectos, en 1987, el PMI publico la primera edición del PMBOK. La edición actual tiene como objetivo esencial proveer de referencias básicas a cualquiera que esté interesado en la gestión de proyectos.

El PMBOK es una colección de procesos y áreas de conocimiento generalmente aceptadas como las mejores prácticas dentro de la gestión de proyectos. Es además un estándar reconocido internacionalmente (IEEE Std. 1490 - 2003) que provee los fundamentos de la gestión de proyectos aplicables a un amplio rango de proyectos.

El PMBOK reconoce cinco procesos básicos y nueve áreas de conocimiento comunes a casi todos los proyectos.

Los procesos se solapan e interactúan a través de un proyecto o fase. Los procesos son descritos en términos de:

- Entradas (documentos, planes, diseños).

- Herramientas y Técnicas (mecanismos aplicados a las entradas).
- Salidas (documentos, productos).

Las nueve áreas del conocimiento mencionadas en el PMBOK son:

1. Gestión de la Integración de Proyectos.
2. Gestión del Alcance en Proyectos.
3. Gestión del Tiempo en Proyectos.
4. Gestión de la Calidad en Proyectos.
5. Gestión de Costos en Proyectos.
6. Gestión del Riesgo en Proyectos.
7. Gestión de Recursos Humanos en Proyectos.
8. Gestión de la Comunicación en Proyectos.
9. Gestión de la Procura (Logística) en Proyectos.

Conclusiones

El análisis de las características expuestas sobre el proceso de pruebas de software nos llevó a la conclusión de que el mismo es de vital importancia a la hora de liberar un software. Ya que define una serie de actividades que garantizan la liberación de un producto final, libre de defectos y que satisfaga las necesidades de los clientes. Se trataron los principales conceptos y definiciones de las pruebas como son: niveles, métodos, técnicas, casos de pruebas.

Luego de analizar estos conceptos se llego a la conclusión que el método de prueba a desarrollar es el de Pruebas de Caja Negra utilizando la Técnica Partición de Equivalencia. Para probar que el sistema funcione como un todo se probara a nivel de Sistema. Apache JMeter es herramienta se seleccionada para desarrollar las Pruebas de Carga y Estrés, ya es versátil, simula usuarios, ahorra recursos, ahorra tiempo y es gratis.

Capítulo 2.
Diseño, Aplicación y Propuesta de Pruebas de Liberación de Software

En el desarrollo de las pruebas de software, para el producto “Sistema de Ayuda Médica para la atención a las Dislipoproteinemias” (alasLIPO), se decidió aplicar una estrategia orientada al comportamiento del sistema, es decir pruebas de caja negra; donde se tienen en cuenta el cumplimiento de los requisitos y otras especificaciones que debe tener dicho producto. Además se decidió aplicar pruebas de Carga y Estrés que *permite resolver problemas de rendimiento de la aplicación, además de anticipar potenciales cuellos de botella en la aplicación o infraestructura que no está bien dimensionada* [15].

En este capítulo se definen elementos básicos relacionados con el diseño, organización y puesta en práctica de las pruebas a ejecutar en el software. Así como la elaboración del Plan de Pruebas y la aplicación de técnicas de diseño de prueba que verifiquen los dominios de entrada y salida, o que descubran errores de funcionalidad comportamiento y rendimiento en el sistema alasLIPO.

2.1 Descripción del Sistema a Probar.

Antes de comenzar a profundizar en el diseño, aplicación y propuesta de pruebas de liberación para este software, debemos conocer ante todo lo que se quiere probar, cómo y las características del producto que se someterá a prueba.

El software “alasLIPO”, es una aplicación web desarrollada sobre el framework Symfony y siguiendo las restricciones de arquitectura de los servidores de INFOMED. Esta aplicación permite gestionar toda la información del proceso de atención de los pacientes dislipidémicos, controlar la evolución de la enfermedad de un paciente dado de forma automatizada, generará automáticamente en cada consulta un diagnóstico y tratamiento para un paciente dado luego de introducir los resultados de sus exámenes y además permitirá realizar al médico reportes sobre los datos de sus pacientes. También cuenta de un foro de discusión que permitirá a cualquier usuario y médicos debatir sobre temas relacionados con las dislipoproteinemias.

alasLIPO constituirá un sistema para apoyar al médico en la asistencia a pacientes que sufren trastornos del metabolismo de los lípidos y las lipoproteínas del plasma. Esta herramienta de trabajo no constituye en manera alguna un sustituto de la labor asistencial del médico, tan solo la complementa.

Se pretende que este sistema sea el primero en Cuba de uso médico para la atención a las Dislipoproteinemias que permita consolidar la información generada en cada terminal por un administrador, con las consecuentes ventajas al crearse una base de datos para análisis estadísticos posteriores. Estos datos enriquecerán la información que el MINSAP pueda necesitar para elaborar programas de prevención cardiovascular, en aras de elevar la calidad de vida de la sociedad cubana.

AI sistema puede accederse a través de la web, hace uso de los principales mecanismos de seguridad a través de la SAAA del sistema informático para la salud cubana, la validación de los datos se realiza del lado del cliente y del servidor. Todas las herramientas de desarrollo utilizadas son software libre.

EI Sistema de Ayuda Médica para la atención a las Dislipoproteinemias garantiza un mejor almacenamiento, búsqueda y procesamiento de datos que posibilitan darles un mejor seguimiento a los pacientes que sufren esta enfermedad.

Dentro de los elementos a probar en este producto, se encuentran todos aquellos que están relacionados con el cumplimiento de los requisitos funcionales como por ejemplo verificar que todos los elementos que lo componen funciones según lo especificado en la documentación, que cada pantalla muestre todos los elementos que la componen o que al pulsar un botón o link estos funcionen arrojando el resultado esperado.

De esta forma se decidió hacer uso de las pruebas de sistema para comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de los sistemas de información con los que se comunica.

2.1.1 Requisitos Funcionales.

Los Requisitos Funcionales constituyen una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, y que tiene que ser alcanzada o poseída por un sistema o

componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

El sistema alasLIPO presenta 28 Requisitos Funcionales que a continuación se relacionan detalladamente:

R1. Validar usuario y contraseña de los usuarios del sistema.
Introducir usuario y contraseña y validar los datos introducidos.

R2. Calcular el valor del Peso ideal.
Calcular y mostrar el valor del peso ideal.

R3. Calcular el valor de las calorías de la dieta.
Calcular y mostrar el valor de las calorías de la dieta.

R4. Calcular el valor del índice de masa corporal.
Calcular y mostrar el valor del índice de masa corporal.

R5. Calcular el valor del riesgo cardiovascular.
Calcular y mostrar el valor de riesgo cardiovascular.

R6. Registrar los datos del paciente.
Guardar los datos del paciente relacionados con sus datos personales, sus antecedentes patológicos familiares, sus antecedentes patológicos personales y los resultados del examen físico, de los síntomas, del lipidograma, de hemoquímica, del ultrasonido dopler modo B de carótidas entre otros.

R6.1 Generar diagnóstico de la primera consulta asociado al paciente.
Luego de introducidos los datos del paciente se genera automáticamente un diagnóstico del paciente asociado a la primera consulta.

R6.2 Generar tratamiento de la primera consulta asociado al diagnostico del paciente.
A partir del diagnóstico generado en la primera consulta asociada al paciente se genera automáticamente el tratamiento del mismo.

R7. Actualizar los datos del paciente.

Se actualizan los datos que no se registraron en la primera consulta o datos que puedan modificársele el valor, como el nombre, primer apellido, segundo apellido, país, provincia, municipio, edad, dirección particular y teléfono.

R8. Buscar paciente según criterios de búsqueda especificados.

Buscar los datos de cada paciente según criterio de búsqueda. Los criterios de búsqueda son nombre, primer apellido, segundo apellido, y carnet de identidad.

R9. Registrar datos de la consulta evolutiva asociada a un paciente.

Guardar los datos del paciente asociados a la consulta evolutiva.

R9.1 Generar diagnóstico de la consulta evolutiva asociado al paciente.

Luego de introducidos los datos del paciente se genera automáticamente un diagnóstico del paciente asociado a la consulta evolutiva.

R9.2 Generar tratamiento de la consulta evolutiva asociado al diagnostico del paciente.

A partir del diagnóstico generado en la consulta evolutiva asociada al paciente se genera automáticamente el tratamiento del mismo.

R10. Visualizar los datos de la historia clínica del paciente.

Visualizar los datos personales del paciente, así como todas las consultas asociadas a un paciente.

R10.1 Generar gráficos del estado del paciente.

Generar gráficos donde se muestre la fecha de cada consulta vs el valor del colesterol, cHDL, cLDL, triglicéridos, índice de masa corporal, presión arterial diastólica y presión arterial sistólica.

R10.2 Visualizar los datos del paciente asociados a las primera consulta.

Visualizar los datos del paciente asociados a la primera consulta.

R10.3 Visualizar los datos del paciente asociados a la consulta evolutiva.

Visualizar los datos del paciente asociados a la consulta evolutiva.

R10.4 Imprimir los datos de la historia clínica del paciente referentes a la primera consulta y a la consulta evolutiva.

Imprimir los datos del paciente asociados a la primera consulta y la consulta evolutiva.

R11. Generar reporte según criterios indicados.

Generar un reporte a partir de datos indicados y seleccionar los datos que se desean ver en el resultado del mismo.

R11.1 Exportar el resultado del reporte generado.

Exportar el resultado del reporte generado a formato Excel.

R11.2 Imprimir el resultado del reporte generado.

Imprimir el resultado del reporte generado.

R12. Insertar un nuevo medicamento.

Insertar los datos de un nuevo medicamento.

R13. Buscar medicamentos existentes.

Buscar los datos de los medicamentos existentes.

R14. Visualizar medicamentos existentes.

Visualizar los datos de los medicamentos existentes.

R15. Eliminar un medicamento.

Eliminar un medicamento existente.

R16. Actualizar los datos de un medicamento.

Actualizar los datos de un medicamento existente.

R17. Insertar un nuevo documento.

Insertar los datos de un nuevo documento.

R18. Buscar documentos existentes.

Buscar los datos de los documentos existentes.

R19. Visualizar documentos existentes.
Visualizar los datos de los documentos existentes.

R20. Eliminar un documento.
Eliminar un documento existente.

R21. Actualizar los datos de un documento.
Actualizar los datos de un documento existente.

R22. Solicitar trasladar un paciente de la consulta.
Solicitar trasladar un paciente para otra consulta.

R23. Trasladar un paciente de la consulta.
Traslada un paciente para otra consulta.

R24. Buscar datos de los usuarios existentes.
Busca los datos de los usuarios que se hayan autenticado en el sistema en algún momento.

R25. Visualizar usuarios existentes.
Visualiza los datos de los usuarios que se hayan autenticado en el sistema en algún momento.

R26. Asignar permisos a los usuarios.
Asigna permiso a los usuarios que se hayan autenticado en el sistema en algún momento para que puedan acceder a la aplicación.

R27. Eliminar un usuario.
Elimina los datos de los usuarios que se hayan autenticado en el sistema en algún momento y no haya realizado consultas a ningún paciente.

R28. Registrar los datos de un usuario.
Registra los datos especificados por el usuario para ubicarlo como pendiente a autorización de acceso al sistema.

2.1.2 Casos de Uso.

Los Casos de Uso como más ampliamente han sido aceptados es como técnica de definición de requisitos. Actualmente se han propuesto como técnica básica del proceso RUP (Kruchten, 1998). Sin embargo, autores como Díez en el 2001, Vilain, Schwabe, Sieckenius, Insfrán, Pastor y Wieringa en el 2002 defienden que pueden resultar ambiguos a la hora de definir los requisitos, por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.

Un Diagrama de Casos de Uso (Use Case Diagram) es una representación gráfica de parte o el total de los actores y Casos de Uso del Sistema, incluyendo sus interacciones.

Todo sistema tiene como mínimo un Diagrama de Caso de Uso del Sistema (Main Use Case), que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso) [16].

Los Requisitos Funcionales del sistema en cuestión han sido agrupados en 15 Casos de Uso, los cuales se relacionan a continuación:

- ✓ **Gestionar Medicamentos**

  Se inserta, actualiza o elimina un medicamento. El sistema registra, actualiza o elimina el medicamento correspondientemente.

- ✓ **Realizar Cálculos**

  Se realizan los cálculos del peso ideal, las calorías de la dieta, el índice de masa corporal y el factor de riesgo cardiovascular.

- ✓ **Generar Reporte**

  Se genera un reporte luego de indicar los criterios para realizarlo y seleccionar los criterios que se mostrarán en el resultado del mismo. Luego se permite exportar a formato Excel o imprimir el resultado del reporte generado.

- ✓ **Registrar Datos del Paciente.**

  Se registran los datos de un nuevo paciente. El sistema muestra la interfaz referente a la inserción de dichos datos, el Médico indica los mismos y entonces el sistema verifica que los datos estén

correctos y sean los necesarios, además el sistema verifica que no exista otro paciente registrado con el mismo carnet de identidad o identidad. Luego de cumplirse todas estas condiciones entonces se guardan los datos asociados a dicho paciente y se genera y registra de ser posible un diagnóstico y tratamiento asociado al paciente.

- ✓ **Buscar Paciente.**

  Para realizar la búsqueda del paciente el Médico indica los criterios de búsqueda Luego el sistema muestra un listado con los pacientes que coincidan con los criterios indicados por el Médico. En caso de que el Médico no indique criterios de búsqueda entonces el sistema muestra todos los pacientes existentes. En ambos casos sólo se muestran los pacientes pertenecientes al Médico en cuestión.

- ✓ **Realizar Consulta Evolutiva.**

  Se muestra la interfaz correspondiente a la Consulta Evolutiva y luego indica los datos asociados a dicha consulta y a partir de los datos indicados se genera un nuevo diagnóstico y tratamiento del paciente. También el Médico puede visualizar los cálculos auxiliares asociados a los datos registrados y/o emitir sus observaciones.

- ✓ **Actualizar Datos del Paciente**

  Se muestra la interfaz correspondiente para actualizar los datos del paciente que faltaban de la primera consulta. Luego del Médico indicar los datos a completar el sistema verifica que estén correctos y en caso positivo los guarda. En caso de que los datos no estén correctos entonces se muestra una alerta.

- ✓ **Visualizar Historia Clínica**

  Se muestra la interfaz correspondiente a la visualización de la Historia Clínica y además brinda la posibilidad de visualizar los datos de todas las consultas realizadas al paciente. También permite imprimir la Historia Clínica del paciente y muestra las gráficas de cómo se comportan ciertos parámetros en el tiempo o en cada consulta que se ha realizado.

- ✓ **Autenticar Usuario**

  El Médico introduce su usuario y contraseña. Luego se verifica que el usuario esté autorizado y en caso positivo el sistema brinda acceso a las funcionalidades según los privilegios del usuario.

- ✓ **Solicitar Traslado de Paciente**

  Se indica buscar los pacientes existentes y luego indica cual es el paciente a solicitar su traslado. El sistema registra la solicitud realizada por el Médico en cuestión.

- ✓ **Trasladar Pacientes**

  El Administrador de sistema indica realizar el traslado de los pacientes pendientes a trasladar para otra consulta.

- ✓ **Buscar Documentos.**

  El Usuario desea buscar los documentos existentes. Luego el sistema muestra una interfaz con el listado de los documentos y sus datos.

- ✓ **Gestionar Documentos.**

  Se desea insertar, actualizar o eliminar un documento. Luego de que el administrador general indique los datos del documento a insertar, actualizar o eliminar entonces el sistema registra, actualiza o elimina el medicamento correspondientemente.

- ✓ **Administrar Usuarios.**

  Se desea visualizar, eliminar un usuario o asignar permisos según el nivel de acceso. Luego el sistema muestra una interfaz con todos los usuarios y da la posibilidad de eliminarlo o asignarle el nivel de acceso. Los usuarios que saldrán en el listado serán según el nivel de acceso del administrador de sistema.

- ✓ **Registrar Usuario**

  El Usuario desea registrarse en el sistema e introduce los datos necesarios para el registro. Luego el sistema registra los datos indicados por el usuario y lo ubica pendiente de autorización para acceso.

2.1.3 Casos de Uso Arquitectónicamente Significativos.

Los Casos de Uso arquitecturalmente significativos son aquellos que representan las partes más críticas de la arquitectura del sistema y demuestran la funcionalidad del sistema [17].

Para alasLIPO los casos de uso significativos han sido priorizados en base al soporte que brindan a las metas del negocio.

Los más significativos son:

- ✓ **Generar Reporte**
- ✓ **Registrar Datos del Paciente.**
- ✓ **Buscar Paciente.**
- ✓ **Realizar Consulta Evolutiva.**
- ✓ **Autenticar Usuario**
- ✓ **Administrar Usuarios.**
- ✓ **Registrar Usuario**

2.2 Plan de Pruebas.

Establecidas las directrices de trabajo, tanto para el equipo de desarrollo como para el equipo de pruebas en el Plan de Aseguramiento de la Calidad del Proyecto (PACP), el equipo de pruebas centrará y coordinará las labores de pruebas desde el Plan de Pruebas.

Para su confección se tienen en cuenta las directrices marcadas en el PACP definiendo los distintos tipos de pruebas que pueden adaptarse al proyecto. El Plan de Pruebas nace, en consecuencia, con labores de diseño en las etapas tempranas del proyecto, donde los requisitos del sistema aportan información crucial para la confección del mismo.

Para garantizar el éxito del plan se desarrollarán las siguientes actividades:

- ✓ Redacción del plan.
- ✓ Obtener aceptación por parte de los implicados.
- ✓ Incluir en la planificación del proyecto la elaboración del plan.
- ✓ Ejecutar y monitorizar su seguimiento.

El objetivo principal de aplicar pruebas al software es ejecutar el programa, habiendo realizado los casos de prueba necesarios con anterioridad, para encontrar la mayor cantidad de fallos en el sistema y corregirlos, de esta manera darle más consistencia al software y una mayor confiabilidad.

Es válido destacar que para realizar un buen caso de prueba debe cumplir dos condiciones fundamentales:

- ✓ Debe poseer una alta probabilidad de encontrar un error que no ha sido detectado hasta el momento.
- ✓ Un caso de prueba correcto es aquel que descubre un error, alcanzando así el éxito.

2.2.1 Propósito del Plan de Pruebas.

Este documento describe el Plan de Pruebas para el sistema alasLIPO. En concreto define los siguientes objetivos específicos:

- ✓ Identifica los elementos que se van a probar.
- ✓ Describe la estrategia de pruebas a seguir en el proceso de prueba.
- ✓ Identifica los recursos necesarios para llevar a cabo el proceso de prueba y estima los esfuerzos que conlleva.
- ✓ Lista los resultados que se obtienen de las actividades de prueba.

2.2.2 Alcance del Plan de Pruebas.

Este Plan de Pruebas describe las pruebas estructurales y funcionales que se aplicarán al sistema de software desarrollado. El objetivo es probar los requisitos definidos en la Especificación de requisitos y en el Modelo de casos de uso.

2.2.3 Métodos a utilizar

Existen diversos métodos para realizar las pruebas de software, entre los más importantes se encuentran la Prueba de Caja Blanca, Prueba de Caja Negra y Prueba de la Estructura de Control.

Las Pruebas de Caja Blanca se centran en verificar que la estructura interna de la unidad sea correcta, las Pruebas de Caja Negra se centran en los requisitos funcionales del software, tratándose desde un enfoque complementario que intenta descubrir diferentes tipos de errores de los métodos de caja blanca, mientras que las Pruebas de la Estructura de Control, por su parte, tienen gran importancia en la garantía de la calidad del software, ampliando la cobertura de la prueba y mejorando la calidad de las pruebas de caja blanca [18].

Por todo lo anteriormente expuesto se utiliza en estas pruebas de liberación el método de Caja Negra, con el fin de estudiar la especificación de las funciones, la entrada y la salida para poder derivar los casos de prueba, definiendo como algo fundamental el probar todas las posibles entradas y salidas del sistema basado en la técnica de Partición de Equivalencia.

Con la aplicación de este método se podrá realizar pruebas a la interfaz del software y examinar aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna

del programa, centrando la atención en el estudio de la especificación del software y las funciones que debe realizar, sus entradas y salidas.

Con la aplicación de estas pruebas al sistema se podrá realizar un buen diseño para los casos de prueba cubriendo todas las entradas y salidas que el mismo debe mostrar y haciendo uso de las técnicas que define este método de prueba.

2.3 Descripción de la Estrategia de las Pruebas realizadas.

Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a una construcción correcta del software. (Fernández, 2002)

Garantizar la calidad del software tiene gran importancia por la implicación en la satisfacción del cliente final. Por esto es necesario lograr definir las actividades y la estrategia que se debe llevar para la evaluación del producto en aras de garantizar la calidad del mismo.

Según Pressman “La estrategia de prueba de software integra un conjunto de actividades que describen los pasos que hay 3eque llevar a cabo de un proceso de prueba: la planificación, el diseño de casos de prueba, la ejecución y los resultados, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software”.

La aplicación de las pruebas y el diseño efectivo de casos de prueba es importante, pero también lo es la estrategia para su utilización y puesta en práctica. En el proyecto, la prueba a veces requiere más esfuerzo que cualquier otra actividad de la Ingeniería del Software. Si se efectúa sin un plan, el tiempo se desaprovecha y el esfuerzo es consumido innecesariamente y, en el peor de los casos, los errores inadvertidos quedarán sin detectar. Por tanto, parece razonable establecer una estrategia sistemática para probar el software.

Para ello es necesario determinar un conjunto de características generales, entre las que podemos mencionar: La prueba comienza por lo pequeño y progresa hacia lo grande. Por esta razón, se debe comenzar las primeras pruebas sobre el componente elemental y aplicar pruebas de caja blanca y de caja negra para descubrir errores en la lógica y la funcionalidad del programa. Después de que los componentes elementales hayan sido aprobados, se procede a su integración. Las pruebas se

efectúan conforme el software se vaya construyendo. Finalmente, una serie de pruebas de alto nivel deben ser ejecutadas una vez que el programa esté totalmente preparado para su operatividad.

En fin, las pruebas deben ser diseñadas para encontrar errores, por lo que una estrategia de prueba debe aislar y probar de forma más concienzuda aquellas partes del software sospechoso de errores.
Es válido e importante tener en cuenta que cuando un software por su tamaño es dividido en módulos las pruebas deben empezar en el análisis de los mismos por separado para acabar en el análisis conjunto del sistema como un todo.

Dentro de los objetivos fundamentales para trazarnos una estrategia de prueba se encuentran:

- ✓ Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de unidad, integración y las pruebas de sistema. Las pruebas de unidad y de integración son necesarias dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- ✓ Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- ✓ Realizar diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Los productos de desarrollo de software en los que se detectan defectos son probados de nuevo y posiblemente devueltas a otra etapa, como diseño o implementación, de forma que los defectos puedan ser arreglados.

Para conseguir estos objetivos el flujo de trabajo de la etapa de pruebas consta de las etapas de planificación, diseño, implementación ejecución y evaluación de las pruebas y además los productos de desarrollo del software fundamentales que se desarrollan en la etapa de pruebas son el plan de pruebas, los casos de prueba, el informe de evaluación de las pruebas, el modelo de prueba que incluye a su vez las clases de prueba, el entorno de configuración de pruebas, componentes de prueba y datos de las pruebas.

La estrategia de pruebas desarrollada para el sistema alasLIPO, entre sus objetivos se define:

- ✓ Identificar los tipos de prueba a utilizar.
- ✓ Definir las técnicas de pruebas que se van a utilizar a lo largo del proceso.
- ✓ Definir el entorno en que se van a desarrollar las prueba.
- ✓ Definir los casos de prueba que se deriven de la aplicación de las técnicas de pruebas.

2.3.1 Método de Pruebas Aplicadas. Técnica Implementada.

Prueba de Caja Negra.

Las Pruebas Funcionales o de Caja Negra son un enfoque para llevar a cabo pruebas, las cuales se derivan de la especificación del programa o componentes. Pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta, y que la integridad de la información externa se mantiene.

Se diseñan casos de prueba para examinar la lógica del programa. Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar casos de prueba que garanticen que se ejercitan todos los caminos independientes de cada módulo, que se ejercitan todas las decisiones lógicas, que se ejecutan todos los bucles, y que se ejecutan las estructuras de datos internas.

Esta prueba tiene como objetivo asegurar la correcta ejecución de todos los caminos lógicos independientes, pudiendo obtener además la complejidad ciclomática del programa.

Técnica de Partición de Equivalencia.

Se decidió utilizar esta técnica porque es muy efectiva a la hora de probar la valides de las condiciones de entrada.

Para la implementación de esta técnica, se comenzó por encontrar las clases de equivalencia, y seguido a esto está la realización de los casos de prueba para las clases válidas en un caso de prueba. Para las no válidas se diseña un caso de prueba por cada clase. Para todas las pruebas realizadas se especifica el posible resultado, comprobando al final la ocurrencia de los errores, comparando el resultado obtenido con el esperado.

Resumiendo, esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Utiliza las clases de equivalencia para el diseño de los casos de prueba, es decir, la entrada de una serie de datos válidos y no válidos, cubriendo en cada caso el máximo número de entradas. Define casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Clase de Equivalencia.

Una *clase de equivalencia* representa un conjunto de estados válidos o no válidos para condiciones de entrada. Normalmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. (Pressman, 2005)

Las clases de equivalencia se definen de acuerdo a las siguientes directrices:

1. Si una condición de entrada específica es un *rango,* se define una clase de equivalencia válida y dos no válidas.
2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
3. Si una condición de entrada requiere un valor específico de entre los de un conjunto, se define una clase de equivalencia válida y una inválida.
4. Si una condición de entrada es lógica (booleana), se define una clase de equivalencia válida y una inválida.

2.3.2 Herramientas de Pruebas para Carga y Estrés utilizadas.

Una buena práctica de software para realizar pruebas a los sistemas de cualquier proyecto de software, cuando estos son de gran magnitud, es su automatización. De lo contrario se volvería muy engorroso y difícil hacer un gran número de pruebas dado el factor tiempo que es muy importante para el desarrollo total del proyecto.

Para la selección de la herramienta adecuada para la puesta en práctica de las Pruebas de Carga y Estrés en el sistema alasLIPO se tuvieron en cuenta las herramientas descritas en el epígrafe 1.8, luego de haber realizado un análisis al respecto se ha decidido utilizar para las pruebas de Carga y Estrés la herramienta JMeter por las funcionalidades que presenta, ser un software libre y gratis.

La utilización del JMeter supone un 95% de tiempo menos para la realización de las pruebas. La mayor inversión de tiempo que se necesita para la realización de las pruebas es la fase de estudio de los casos de uso críticos en la aplicación y la elaboración del plan de pruebas. Permite almacenar los resultados de la prueba y generar gráficos que representan los aspectos que se han probado.

La posibilidad de contar con material para el estudio de la utilización de la herramienta, supone una garantía en el éxito de la ejecución de las pruebas. Otro aspecto a favor de esta herramienta es su desarrollo en el marco del software libre, política que apoya la universidad y que por las características

existe un por ciento superior de confianza en la utilización de la misma ya que se puede contar con todo el código que genera a la herramienta.

Se considera importante exponer que para las pruebas de carga y estrés automatizadas por JMeter se define 5 importantes etapas en las que se enmarca esta descripción (Fig. 2.1).

Fig. 2.1 Proceso de automatización de las Pruebas de Carga y Estrés.

Con la aplicación de las Pruebas de Carga y Estrés a través de la herramienta seleccionada se emplearía menos tiempo, menos recursos y se obtendría un producto de mayor calidad.

2.4 Estrategia de Trabajo.

La estrategia de trabajo para la aplicación de pruebas de software al sistema alasLIPO siguió las pautas y objetivos trazados en la descripción de la Estrategia de Prueba descrita anteriormente en el epígrafe 2.4. A continuación se describirá de forma muy rápida para evitar caer en redundancia como se siguió el trabajo durante la aplicación de las pruebas al sistema alasLIPO.

2.4.1 Plan de Prueba.

Control de Versiones

Fecha	*Versión*	*Descripción*	*Autor*
01/03/09	*1.0*	*Elaboración del plan de pruebas.*	*Yunior Mesa Reyes* *Yuniel Torres*
18/04/09	*1.0*	*Actualización de cronograma y diagramas.*	*Yunior Mesa Reyes*
28/05/09	*1.0*	*Actualización de cronograma.*	*Yunior Mesa Reyes*

2.4.1.1 Propósito

El plan de pruebas del proyecto alasLIPO consiste en seleccionar y coordinar todas las actividades para garantizar la calidad del producto durante el ciclo de vida del proyecto. En el mismo se explica alcance, requerimientos a probar, estrategia a seguir, recursos requeridos, cronograma y herramienta a utilizar.

2.4.1.2 Alcance.

Este documento describe el Plan de Pruebas para el Sistema de Ayuda Médica para la atención a las Dislipoproteinemias **alasLIPO**.

Concretamente persigue:

- Identificar los elementos que se van a probar.
- Describir la estrategia de pruebas a seguir en el proceso de prueba.
- Definir los recursos necesarios para llevar a cabo el proceso de prueba y estimar los esfuerzos que conlleva.
- Listar los resultados que se obtienen de las actividades de prueba

2.4.1.3 Referencias.

Código	Título
1	Documentos Especificación de los Requisitos del Sistema
2	Documentos Modelo de Casos de Uso del Sistema
3	Documentos Diseño de Casos de Prueba

2.4.1.4 Estrategia de Prueba

En esta sección se describe la forma en que se realizarán las pruebas al software, identificándose los niveles, técnicas y métodos empleados.

El nivel por el cual se realizarán las pruebas de liberación al producto será:

- Nivel de Prueba de Sistema.

En aras de aclarar la razón por la cual solamente se probará en el nivel mencionado, cabe destacar que en la etapa del ciclo de vida en que se encuentra el software, ya se ha transitado por los niveles precedentes, excepto el nivel de Pruebas de Integración ya que nuestro software posee un solo

módulo. Se han realizado las Pruebas de Desarrolladores e Independientes, de modo que el sistema necesita ser probado íntegramente y luego comprobar que funcione como un todo.
Los pasos a seguir para el flujo de trabajo desarrollado son los siguientes:

- ✓ Realización de un plan de pruebas.
- ✓ Definición de los niveles de pruebas en los cuales se va a probar.
- ✓ Definición de las técnicas de pruebas empleadas en los niveles utilizados.
- ✓ Aplicación de listas de chequeos para la documentación.
- ✓ Realización de pruebas exploratorias al software.
- ✓ Realización de diseños de casos de pruebas.
- ✓ Aplicación del método de caja negra para cada caso de prueba diseñado.
- ✓ Elaboración de la plantilla de No Conformidades.
- ✓ Realización de pruebas de carga y estrés.

Objetivo

Comprobar que todos los requerimientos funcionales del software se comporten de acuerdo a lo esperado o cómo está documentado.

Técnicas

El conjunto de pruebas a realizar se compondrá por Pruebas de Funcionalidad, Pruebas de Seguridad, Pruebas de Confiabilidad y Pruebas de Rendimiento con sus consiguientes tipos de pruebas.

Mediante las Pruebas Funcionales se tratará de garantizar que el software trabaje funcionalmente como se especificó por el cliente. Con las Pruebas de Seguridad se tratará de verificar que un usuario solo pueda acceder a las funciones y datos que tiene permitido, o sea, que para cada tipo de usuario conocido, las funciones, datos apropiados y todas las transacciones funcionan como se esperaba.

Las Pruebas de Volumen se evidencian implícitamente dentro de las Pruebas de Carga y Estrés, estas últimas contenidas dentro de las Pruebas de Confiabilidad y Rendimiento, pretendiendo comprobar el desempeño, resistencia y rendimiento del sistema. Tácitamente cuando se prueba la funcionalidad del sistema empleando también de forma implícita la Destrucción Aleatoria, si se es riguroso en la realización de estas pruebas, se verifica también la fiabilidad, la seguridad, la capacidad de carga y volumen y el rendimiento del sistema.

Métodos

Se implementará para las pruebas el método de Caja Negra y dentro de éste, la técnica de Partición de Equivalencia. Dicha selección se basa en las características de ambos métodos y las técnicas que lo soportan, arrojando cada cual por su parte, diferentes resultados, pero en esencia, se busca mayor eficacia y eficiencia a la hora de encontrar defectos y verificar la calidad, de forma tal que se minimicen tiempos, costo y se obtenga un resultado satisfactorio.

Entorno de Prueba

El sistema podrá ser usado sobre los sistemas operativos Windows y Linux.

- Para el servidor de aplicación:
    - Pentium 3.00 GHz o superior.
    - 0.99 GB de RAM
- Conexión TCP/IP

Para automatizar las pruebas de Carga y Estrés se requirió además:

- Software: Microsoft Word.
- Herramienta: JMeter
- Máquina Virtual: Java Virtual Machine 1.3 o superior.
- Navegador: Internet Explorer

Proceso

La ejecución del proceso de pruebas está basada en el procedimiento establecido por el Grupo Laboratorio de Pruebas de la Dirección de Calidad. Es importante destacar que este procedimiento no es un proceso documentado sino que constituye simplemente una buena práctica para la ejecución de las pruebas de Liberación y Aceptación, resultado de la experiencia acumulada por los especialistas de la dirección. Actualmente se trabaja en la definición de los procesos del Laboratorio Industrial de Pruebas de Software, con la creación del Centro CaliSoft.

Durante este proceso de liberación del software se realizan pruebas funcionales para validar cuando el comportamiento observado del software cumple o no con sus especificaciones. Dichas pruebas se realizan a todo el sistema. Este proceso de liberación contiene las pruebas al software y la revisión técnica de la documentación del software, argumento importante a tener en cuenta al momento de evaluar la calidad de un producto de software.

El proceso está determinado por las siguientes actividades:

- *Reunión de Inicio.*
- *Entrega del producto.*
- *Ejecutar Revisiones Técnicas a la documentación.*
- *Ejecutar Pruebas Funcionales a la aplicación.*
- *Elaborar Documento de No Conformidades por CU.*
- *Informar resultado de las pruebas.*
- *Revisar respuestas de las No Conformidades.*
- *Ejecutar pruebas de Carga y Estrés.*
- *Evaluación del Producto.*
- *Liberación del Producto.*

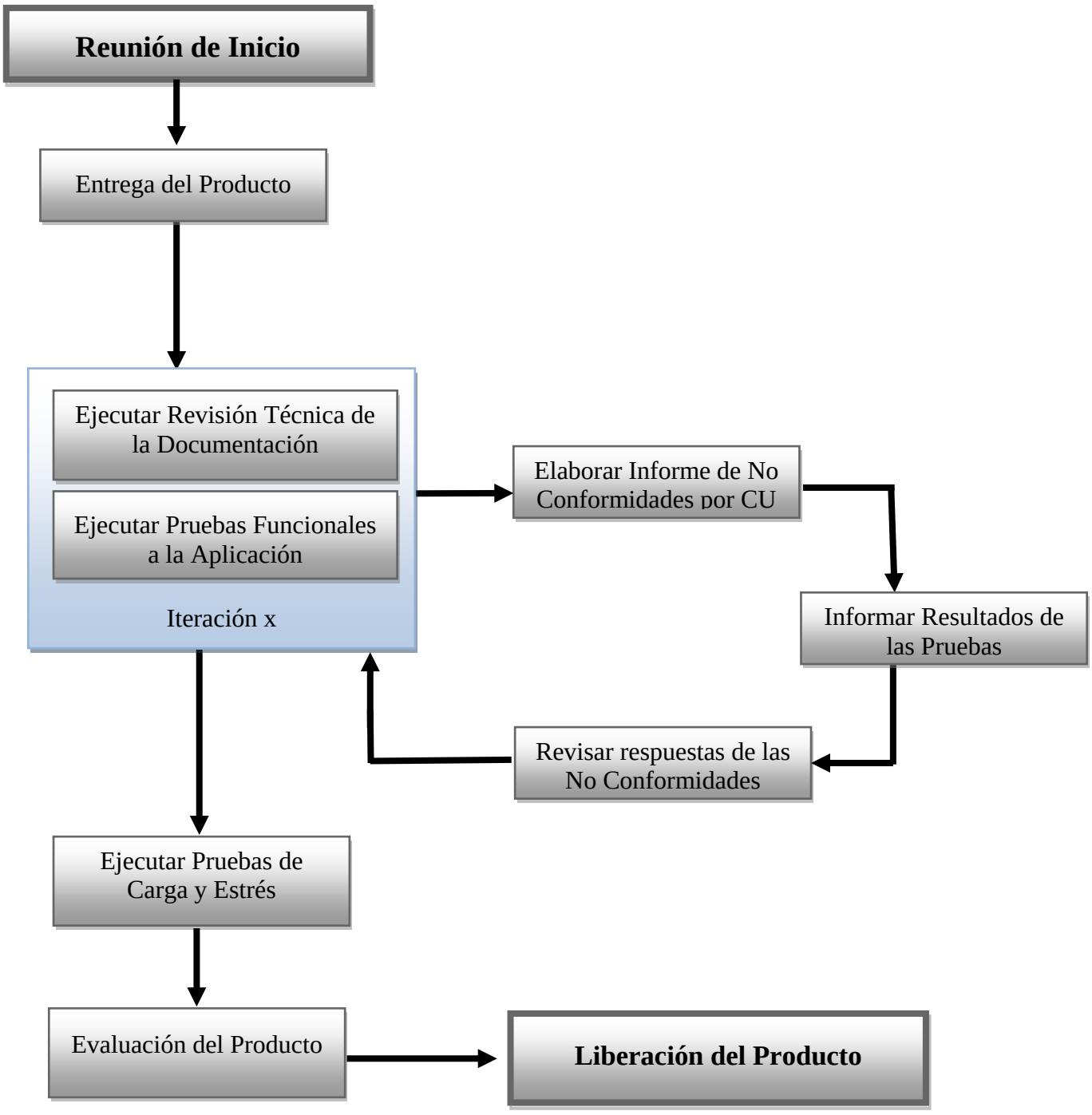

Fig. 2.2 Proceso de Pruebas de Liberación de alasLIPO.

- ✓ Reunión de Inicio

Se concilia y acepta el Plan de Pruebas de Liberación entre las partes involucradas. Se identifican los riesgos y se actualiza el cronograma.

- ✓ Entrega del Producto

Se realiza la entrega del producto por parte del Equipo de Desarrolladores.

- ✓ Ejecutar Revisiones Técnicas a la documentación

Se realiza una revisión de la documentación teniendo en cuenta los elementos técnicos definidos por la metodología de desarrollo, la correspondencia con lo documentado y lo implementado, y los errores de ortografía y redacción, por iteraciones independientes. Estas revisiones se realizan por iteraciones independientes hasta cumplir con los criterios de término.

- ✓ Ejecutar Pruebas Funcionales a la aplicación

Se ejecutan las Pruebas Funcionales por iteraciones independientes, a partir del método de prueba de Caja Negra con la técnica de Partición de Equivalencias.

- ✓ Elaborar Informe de No Conformidades por CU.

A partir de la culminación de una iteración determinada se elabora el Documento de No Conformidades con los resultados de las pruebas y revisiones.

- ✓ Informar Resultados de las Pruebas

Se informa al equipo de desarrollo cuáles fueron los defectos detectados enviándoles el Documento de No Conformidades.

- ✓ Revisar respuestas de las No Conformidades

Se revisan la resolución de las No Conformidades en la nueva versión del documento o de la aplicación a partir de las respuestas de los desarrolladores. Se realizan Pruebas de Regresión para verificar que no se hayan introducido nuevos errores.

- ✓ Ejecutar Pruebas de Carga y Estrés

Una vez concluidas todas las pruebas y revisiones, se ejecutan las pruebas de carga y estrés con la herramienta JMeter y se evalúan los diferentes parámetros de calidad, dígase Usabilidad, Seguridad, Confiabilidad y Portabilidad.

- ✓ Evaluación del Producto

Se realiza la evaluación de la aplicación teniendo en cuenta los atributos de calidad que establece la Norma ISO 9126.

- ✓ Liberación del Producto

Se declara el producto Liberado, a partir de la liberación de la documentación y de la aplicación.

Casos de Prueba

Se elaborarán tantos Casos de Prueba como Casos de Uso tenga el sistema. Dadas las características del proyecto y la necesidad de presentar la mayor cantidad de artefactos a uno de los cortes de tesis, se decidió comenzar por los Casos de Uso Arquitectónicamente Significativos y continuar luego con los restantes. El diseño de Casos de Pruebas constituye un proceso paralelo al periodo de ejecución de las Pruebas de la Aplicación. Durante este tiempo se diseñaron y se refinaron constantemente, en función de detectar la mayor cantidad de defectos posibles.

Criterios de Término

El criterio a tener en cuenta para culminar con las pruebas de liberación está determinado por la resolución del 100 % de las No Conformidades y la completitud de la revisión de la documentación y de las pruebas a la aplicación.

Herramientas

Posterior a todo el análisis realizado, se concluye con la selección de JMeter, herramienta dinámica que además de evaluar el rendimiento mediante las pruebas de Carga y Estrés, utiliza el método de Caja Negra, probando todas las funcionalidades del software, y proveyendo de esta manera una retroalimentación y reforzamiento en las pruebas funcionales ya realizadas con anterioridad. Además se clasifica por su especialización en una herramienta de grabación y reproducción. Se empleará la versión 2.3 de JMeter, provista por Jakarta.

2.4.1.5 Plan de Proyecto.

	Task Name	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	Reunión de Inicio	1 día	mar 03/02/09	mar 03/02/09		Equipo de Desarrolladores,Yunior,Yuniel
2	Entrega de la Documentación	1 día	mié 04/02/09	mié 04/02/09	1	Equipo de Desarrolladores
3	Revisión Técnica de la Documentación	16 días	jue 05/02/09	jue 26/02/09	2	Yunior,Yuniel
4	Capacitación	1 día	jue 05/02/09	jue 05/02/09	1	Yunior,Yuniel,Equipo de Desarrolladores
5	Diseño de los Casos de Prueba	9 días	mar 17/02/09	vie 27/02/09	2	Yunior,Yuniel
6	Entrega de la Aplicación	1 día	vie 13/02/09	vie 13/02/09	1	Equipo de Desarrolladores
7	**Pruebas de la Aplicación**	**11 días**	**lun 16/02/09**	**lun 02/03/09**	**6**	
8	Pruebas Exploratorias	1 día	lun 16/02/09	lun 16/02/09	6	Yunior,Yuniel
9	Pruebas Funcionales	9 días	mar 17/02/09	vie 27/02/09	8	Yunior,Yuniel
10	Pruebas de Carga y Estrés	1 día	sáb 28/02/09	lun 02/03/09	9	Yunior,Yuniel
11	Evaluación de Atributos de Calidad del Producto	1 día	sáb 28/02/09	lun 02/03/09	9	Yunior,Yuniel
12	Liberación del Producto	1 día	lun 02/03/09	lun 02/03/09	9,10,11,3	Yunior,Yuniel,Equipo de Desarrolladores

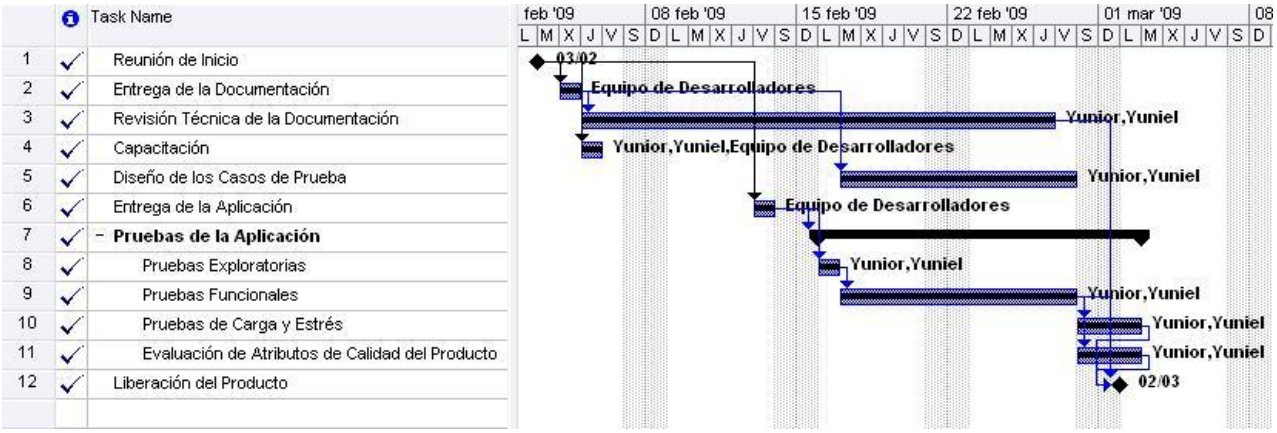

Fig. 2.3 Cronograma de Trabajo del Proceso de Liberación de alasLIPO.

2.4.2 Niveles y Técnicas.

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Teniendo en cuenta esto, las pruebas se agrupan por niveles, de acuerdo con las diferentes etapas del proceso de desarrollo:

- ✓ **Prueba de Desarrollador**
- ✓ **Prueba Independiente**
- ✓ **Prueba de Unidad**
- ✓ **Prueba de Integración**
- ✓ **Prueba de Sistema**
- ✓ **Prueba de Aceptación**

Por su parte las técnicas pueden concentrarse en un conjunto de habilidades correctivas agrupadas en diferentes categorías.

En el caso del sistema alasLIPO las pruebas se desarrollaron a nivel de Sistema, realizadas ya cuando el software estuvo funcionando como un todo.
Con esta actividad las pruebas fueron dirigidas a verificar el programa final, con los componentes de software y hardware ya integrados.

2.4.3 Revisión de la Documentación.

Durante el proceso de revisión técnica de la documentación (Ver Fig. 2.2) relacionada con la aplicación, se sometieron a una rigurosa revisión los siguientes documentos:

- ✓ Roles y Permisos
- ✓ Modelo de CUS
- ✓ Manual de Usuario
- ✓ Especificación de Requisitos
- ✓ Diccionario de Datos

Cabe señalar que fue necesario realizar hasta una sexta iteración para dos de los documentos pertenecientes al expediente del proyecto y que el tiempo de respuesta en todos los casos por parte de los desarrolladores fue de 2 a 3 días.

Dentro de los aspectos que se tuvieron en cuenta durante la realización de la revisión técnica a la documentación se encuentran:

- ✓ Estructura de los documentos.
- ✓ Concordancia gramatical y ortografía.
- ✓ Claridad de las ideas en la redacción.
- ✓ Formato y Contenido.
- ✓ Concordancia del número de página que aparece en el índice con el contenido que se refleja realmente en dicha página.
- ✓ Concordancia entre la descripción de la documentación y lo real en la aplicación.
- ✓ Coincidencia del total de páginas que aparecen en las reglas de confidencialidad con el total de páginas real que tiene el documento.

Dentro de los principales problemas encontrados se encuentran:

- ✓ Errores de concordancia y ortografía.
- ✓ Errores de contenido.
- ✓ Errores de formato.
- ✓ Errores de concordancia entre la descripción de la documentación y lo real en la aplicación.
- ✓ Errores de numeración y coincidencia de las páginas contra el índice y las reglas de confidencialidad.

Nombre	V1.0	V 1.1	V1.2	V1.3	V1.4	V1.5	V1.6		TOTAL
Roles y Permisos	2 NC (1s – 1 ns)	LIBERADO							2
	2 R								2
Modelo de CUS	35 NC (22s – 13 ns)	1 NC (1s – 0 ns)	1 NC (1s – 0 ns)	1 NC (1s – 0 ns)	1 NC (1s – 0 ns)	4 NC (4s – 0 ns)	2 NC (2s – 0 ns)	LIBERADO	45
	9 R	0 R	0 R	0 R	1 R	0 R	0 R		10
Manual de Usuario	12 NC (9s – 3 ns)	2 NC (1s – 1 ns)	1 NC (1s – 0 ns)	LIBERADO					15
	10 R	0 R	1 R						11
Especificación de Requisitos	13 NC (12s – 1 ns)	2 NC (2s – 0 ns)	3 NC (3s – 0 ns)	1 NC (1s – 0 ns)	LIBERADO				19
	9 R	2 R	0 R	1 R					12
Diccionario de Datos	4 NC (4s – 0 ns)	2 NC (2s – 0 ns)	1 NC (1s – 0 ns)	LIBERADO					7
	1 R	0 R	0 R						1
TOTAL	66 NC (48s – 18 ns)	7 NC (6s – 1 ns)	6 NC (6s – 0 ns)	2 NC (2s – 0 ns)	1 NC (1s – 0 ns)	4 NC (4s – 0 ns)	2 NC (2s – 0 ns)	88 NC (69s – 19 ns)	88 NC (69s – 19 ns)
	31 R	2 R	1 R	1 R	1 R	0 R	0 R	36 R	36 R

NC: No Conformidad **s:** significativa **ns:** no significativa **R:** Recomendación

Fig. 2.4 Cierre de Liberación de la Documentación del sistema alasLIPO.

2.4.4 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 [24].

James Bach define como Pruebas Exploratorias al proceso simultáneo de exploración del producto (aprendizaje), diseño y ejecución de pruebas [25].

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.

“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" [26].

Las Pruebas Exploratorias son el resultado del estudio y aplicación del sistema en el contexto adecuado. Constituyen una guía para las pruebas venideras en base a los resultados que se van obteniendo durante su realización.

2.4.5 Diseños de Casos de Prueba.

Los casos de prueba especifican una forma de probar el sistema, incluyendo la entrada y salida con la que se ha de probar y las condiciones bajo las que ha de probarse. Se describen las clases válidas e inválidas que se pueden utilizar para realizar las pruebas del software. Su propósito es identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de la prueba son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (casos de uso).

Para generar los casos de prueba a partir de un caso de uso totalmente detallado se describe un proceso de tres pasos:

1. Para cada caso de uso, generar un sistema completo de escenarios.
2. Para cada escenario, identificar por lo menos un caso de prueba y las condiciones que hagan que "se ejecute. “
3. Para cada caso de prueba, identificar los valores de los datos con los cuales se hará la prueba.

Para generar los escenarios se lee la descripción textual del caso de uso y se identifican todas las combinaciones posibles de caminos de ejecución del caso de uso, es decir todas las combinaciones posibles entre el camino principal y los caminos alternativos y se le asigna un nombre. Cada combinación será un escenario del caso de prueba.

Un escenario es un "camino" completo a través del caso de uso. Los usuarios finales del sistema pueden seguir muchos caminos cuando ejecutan la funcionalidad especificada en el caso de uso. Siguiendo el flujo básico sería un escenario. Siguiendo el flujo básico más el alternante flujo 1A sería otro. El flujo básico más el alternante flujo 2A sería un tercero, y así sucesivamente.

Por su parte las secciones son los caminos excluyentes dentro del Caso de Uso (que tributa a los Casos de Prueba), acciones independientes que puede ejecutar el actor. Una misma sección puede tener diferentes escenarios.

Es importante destacar la necesidad de confeccionar un buen caso de prueba, pues de esta forma se detectarán la mayor cantidad de errores posibles, permitiendo con esto que el software llegue prácticamente libre de defectos, a las manos del usuario final.

Para alasLIPO se diseñaron los casos de pruebas según los 15 casos de uso que se definieron en la solución propuesta, teniendo en cuenta sus requisitos.
Dadas las características del proyecto y la necesidad de presentar la mayor cantidad de artefactos a uno de los cortes de tesis, se decidió comenzar por los Casos de Uso Arquitectónicamente Significativos y continuar luego con los restantes. El diseño de Casos de Pruebas constituye un proceso paralelo al periodo de ejecución de las Pruebas de la Aplicación. Durante este tiempo se diseñaron y se refinaron constantemente, en función de detectar la mayor cantidad de defectos posibles.

Una muestra de este diseño se puede ver reflejada en el Diseño de Casos de Pruebas Realizar Cálculos.

Inicialmente se procede a definir las secciones a probar, tiendo en cuenta los siguientes aspectos: Nombre de la Sección, Escenarios de la Sección, Descripción de la Funcionalidad a través de una breve descripción de la funcionalidad que se va a probar, así como el Flujo Central donde se describen paso a paso las acciones del usuario para ejecutar la prueba.

Posteriormente se pasa a realizar la descripción de las variables para luego llenar la tabla correspondiente a las variables donde se dan valores de entradas válidas, inválidas y no aplican, con el objetivo de comprobar la veracidad de los datos de entrada descritos anteriormente en las diferentes secciones y escenarios. En caso de resultar insatisfactorio el resultado que arrojen éstas, se procede a llenar la tabla de registro de las No Conformidades encontradas para hacérselas llegar al equipo de desarrollo para su posterior revisión.

Un ejemplo de escenarios, secciones y variables se muestra a través del Diseño de Caso de Prueba Realizar Cálculos:

✓ ***Secciones a probar.***

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC1: Realizar Cálculos	EC1.1: Realizar Cálculos.	El sistema muestra una interfaz para Calcular calorías de la dieta o el valor del riesgo cardiovascular	*Pasos a desarrollar para probar la funcionalidad que se indicó.*
SC2: Calcular calorías de la dieta.	EC 2.1: Calcular Calorías de la dieta.	El sistema muestra los resultados correspondientes al cálculo de las calorías de la dieta.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*
	EC 2.2: Imprimir dieta.	El sistema imprime la dieta dando clic en la imagen correspondiente.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*
	EC 2.3: (F - A) Calcular Calorías de la dieta, si el usuario dejó campos vacios	Se emite un mensaje de alerta.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*
	EC 2.4: (F - A) Calcular Calorías de la dieta, si el usuario indico valores incorrectos.	El sistema emite un mensaje de alerta.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*
	EC 2.5: (F - A) No desear imprimir dieta.	Se sale de la sección.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*
SC3: Calcular valor del Riesgo cardiovascular	EC 3.1: Calcular valor del Riesgo cardiovascular.	El sistema muestra la interfaz correspondiente para calcular el valor del riesgo cardiovascular, el usuario introduce los datos, el sistema los verifica realiza el cálculo y muestra el resultado correspondiente a dicho cálculo.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*
	EC3.2: (F - A) Calcular valor del Riesgo cardiovascular si el usuario dejo campos vacios.	Si el usuario dejo campos vacios el sistema emite un mensaje de alerta.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*

	EC3.3: (F - A) Calcular valor del Riesgo cardiovascular si el usuario indico valores incorrectos.	Si el usuario indicó valores incorrectos, el sistema emite un mensaje de alerta.	*Pasos a desarrollar para probar la funcionalidad que se indicó.*

- ✓ ***Descripción de la Variable.***

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Edad	Texto Numérico	N	Edad del Paciente.
2	Sexo	Lista de selección	N	Sexo del Paciente.
3	¿Es Fumador?	Lista de selección	N	Paciente fuma o no.
4	Colesterol	Texto Numérico y Lista de selección	N	Valor del colesterol del paciente.
5	cHDL	Texto Numérico y Lista de selección	N	Valor del cHDL del paciente.
6	Presión Arterial Sistólica	Texto Numérico	N	Valor de la Presión Arterial Sistólica del paciente.
7	¿Cumple tratamiento para la Hipertensión?	Lista de selección	N	Cumple tratamiento o no el paciente.
8	Peso	Texto Numérico	N	Peso del paciente.
9	Talla	Texto Numérico	N	Talla del paciente.

- ✓ ***Matriz de Datos.***

Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

SC 1: < Realizar Cálculos >

Id del escenario	Escenario	Calcular Calorías de la dieta.	Calcular riesgo cardiovascular	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Realizar Cálculos.	V	N/A	El sistema muestra la interfaz correspondiente a Calcular Calorías de la Dieta.	**SATISFACTORIO**
		N/A	V	El sistema muestra la interfaz correspondiente a Calcular Riesgo Cardiovascular.	**SATISFACTORIO**

SC 2: < Calcular calorías de la dieta. >

Id del escenario	Escenario	Variables			Respuesta del Sistema	Resultado de la Prueba
		8	9	Imprimir		
EC 2.1	Calcular Calorías de la dieta.	V	V	N/A	El sistema muestra el resultado del índice de masa corporal del peso ideal y de las calorías de la dieta correspondientes al paciente.	**SATISFACTORIO**
EC 2.2	Imprimir Dieta	N/A	N/A	V	El sistema imprime la dieta.	**SATISFACTORIO**
EC 2.3	(F - A) Calcular Calorías de la dieta si el usuario dejó campos vacios.	I	V	N/A	El sistema emite un mensaje para que llene los campos obligatorios.	**SATISFACTORIO**
		V	I	N/A		
EC 2.4	(F - A) Calcular Calorías de la dieta si el usuario	I	V	N/A	El sistema emite un mensaje de alerta.	**SATISFACTORIO**

	indico valores incorrectos.	V	I	N/A		
EC 2.5	(F - A) No desear imprimir dieta.	N/A	N/A	N/A	Se sale de la sección.	**SATISFACTORIO**

SC 3: < Calcular valor del Riesgo cardiovascular >

Id del escenario	Escenario	Variables							Respuesta del Sistema	Resultado de la Prueba
		1	2	3	4	5	6	7		
EC 3.1	Calcular valor del Riesgo cardiovascular.	V	V	V	V	V	V	V	El sistema los verifica, realiza el cálculo y muestra el resultado correspondiente a dicho cálculo.	**SATISFACTORIO**
EC 3.2	(F - A) Calcular valor del Riesgo cardiovascular si el usuario dejo campos vacios.	I	V	V	V	V	V	V	El sistema emite un mensaje para que llene los campos obligatorios.	**SATISFACTORIO**
		V	I	V	V	V	V	V		
		V	V	I	V	V	V	V		
		V	V	V	I	V	V	V		
		V	V	V	V	I	V	V		
		V	V	V	V	V	I	V		
		V	V	V	V	V	V	I		
EC 3.3	(F - A) Calcular valor del Riesgo cardiovascular si el usuario indico valores incorrectos.	I	V	V	V	V	V	V	El sistema emite un mensaje de alerta.	**SATISFACTORIO**
		V	I	V	V	V	V	V		
		V	V	I	V	V	V	V		
		V	V	V	I	V	V	V		
		V	V	V	V	I	V	V		
		V	V	V	V	V	I	V		
		V	V	V	V	V	V	I		

2.4.6 Caja Negra. Partición de Equivalencia.

Dentro de la Estrategia de Trabajo se define el método de prueba a emplear, así como todas sus especificidades descritas detalladamente en el epígrafe 2.3.1 de este mismo capítulo.

Solo cabe acotar que cuando que estas pruebas se llevan a cabo sobre la interfaz del software, por lo que 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 una salida correcta, así como que la integridad de la información externa se mantiene.

Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

2.4.7 No Conformidades de la Aplicación.

Durante el proceso de revisión de la aplicación, ésta se sometió a una rigurosa revisión para la detección de No Conformidades, reflejadas en la plantilla existente como artefacto del mismo (Ver Anexo # 1). En este proceso, como se muestra en la siguiente tabla (Fig. 2.3), se llegó hasta una versión 1.1 en uno de los diseños realizados.

Dentro de los principales problemas encontrados se encuentran:

- ✓ Errores de validación de datos.
- ✓ Errores de concordancia entre la interfaz implementada y la descripción de la documentación.
- ✓ Errores de concordancia entre las funcionalidades implementadas del sistema y la descripción de la documentación.

Nombre	V1.0	V 1.1	TOTAL
DCP Buscar Paciente	1 NC (1s – 0 ns)		1 NC (1s – 0 ns)
	0 R		0 R
DCP Generar Reporte	3 NC (3s – 0 ns)	1 NC (1s – 0 ns)	4 NC (4s – 0 ns)
	0 R	0 R	0 R
DCP Gestionar Medicamento	3 NC (3s – 0 ns)		3 NC (3s – 0 ns)
	0 R		0 R
DCP Trasladar Paciente	1 NC (1s – 0 ns)		1 NC (1s – 0 ns)
	0 R		0 R
DCP Gestionar Documentos	2 NC (2s – 0 ns)		2 NC (2s – 0 ns)
	0 R		0 R
DCP Autenticar Usuario	1 NC (1s – 0 ns)		1 NC (1s – 0 ns)
	0 R		0 R
DCP Registrar Usuario	1 NC (1s – 0 ns)		1 NC (1s – 0 ns)
	0 R		0 R
TOTAL	**12 NC (12s – 0 ns)**	**1 NC (1s – 0 ns)**	**13 NC (13s – 0 ns)**
	0 R	**0 R**	**0 R**

NC: No Conformidad **s:** significativa **ns:** no significativa **R:** Recomendación

Fig. 2.5 Cierre de Liberación de la aplicación del sistema alasLIPO.

2.4.8 Prueba de Carga y Estrés.

Existe actualmente un sinnúmero de herramienta automatizadas para la ejecución de las pruebas de software, pudiendo citar algunas como Rational Test y JMeter, entre muchas otras. Hasta hace un tiempo atrás, las pruebas realizadas al software en la Universidad de las Ciencias Informáticas se han hecho de forma manual.

Hoy en día se han estado automatizando estas herramientas, haciendo más rápido y con mayor calidad el desarrollo de estas pruebas. En el caso particular del sistema alasLIPO la herramienta seleccionada es JMeter por todas las funcionalidades que presenta, descritas en el epígrafe 1.8, retomadas en el epígrafe 2.3.2 y que se resumen en la figura 2.4 que a continuación se presenta.

Fig. 2.6 Funcionalidades de la herramienta para Carga y Estrés. JMeter.

2.5 Diseños de Casos de Pruebas

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular ó una función esperada (Ver Anexo # 2). La entidad más simple siempre es ejecutada como una unidad, desde el comienzo hasta el final, estos casos de pruebas deben verificar:

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

Formalmente, los casos de prueba escritos contienen principalmente tres partes:

1. Introducción/visión general, contiene información general acerca los casos de prueba.
2. Actividad de los casos de prueba.
3. Resultados.

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

Los siguientes son casos de pruebas comunes:

1. Un caso de prueba que especifica como probar un caso de uso o un escenario específico de un caso de uso. Un caso de prueba de este tipo incluye una verificación del resultado de la interacción entre los actores y el sistema, que se satisfacen las precondiciones y pos-condiciones específicas por el caso de uso y que se sigue la secuencia de acciones especificadas por el caso de uso. Un caso de prueba basado en un caso de uso especifica típicamente una prueba del sistema como “caja negra”, es decir, una prueba del comportamiento observable externamente del sistema.

2. Un caso de prueba que especifica como probar una realización de caso de uso-diseño o un escenario específico de la realización. Un caso de prueba de este tipo puede incluir la verificación de la interacción entre los componentes que implementan dicho caso. Los casos de pruebas basados en una realización de caso de uso típicamente especifican una prueba del sistema como “caja blanca”, es decir, una prueba de la interacción interna entre los componentes del sistema [27].

Conclusiones

En este capítulo se describe muy brevemente el sistema alasLIPO, al que se le diseñó pruebas de caja negra con el objetivo de detectar errores para su posterior corrección.

Las pruebas se diseñan para comprobar que existen errores, no para demostrar que no existen. Además quedaron explicadas las características que tendrá el Plan de Pruebas diseñado para el sistema alasLIPO, la estrategia a seguir en el diseño de los casos de pruebas y la configuración del entorno que se empleará en todo la etapa de pruebas.

También se obtuvieron todos los procedimientos de pruebas, los cuales permitieron una mejor comprensión del funcionamiento del sistema, pudiendo utilizar los mismos para el diseño y ejecución de los casos de pruebas.

Para realizar una exitosa liberación a la documentación y a la aplicación se tuvo en cuenta los parámetros definidos en las Listas de Chequeo para luego emitir una evaluación correspondiente.

Capítulo 3.
Análisis de los Resultados y Valoración de la Propuesta.

La evaluación de pruebas recoge, organiza, y presenta los resultados obtenidos en el desarrollo de las mismas. En el proceso de pruebas es muy importante el registro de la documentación pertinente.

La realización de una evaluación de los resultados obtenidos contra los esperados, a partir de los entregables establecidos, donde se expongan cada uno de los detalles de las pruebas así como los errores que fueron detectados luego de la aplicación de las pruebas forma parte de las actividades a realizar durante el proceso de prueba y liberación de un software por muy sencillo que este sea.

Este capítulo hará referencia al análisis de los resultados obtenidos luego de diseñar y aplicar el proceso de pruebas al Sistema alasLIPO.

3.1 Análisis de los resultados obtenidos en las pruebas de Caja Negra.

Las pruebas se ejecutaron utilizando una PC con sistema operativo Microsoft Windows XP Profesional Service Pack 2, con 1GB de memoria RAM y microprocesador Pentium IV. Se instaló la aplicación web en la PC y se probaron todas las funcionalidades implementadas, las cuales requerían conexión a una base de datos que se encontraba en la propia PC.

Las pruebas fueron realizadas por dos probadores que no pertenecen al equipo de desarrollo del sistema. Los resultados de los errores comprenden los de los casos diseñados para las entradas, los errores de interfaz y de las pruebas de carga y estrés.
Por cada caso de uso se realizó un caso de prueba, con datos válidos y datos no válidos, obteniéndose un resultado en cada una de las pruebas que se ejecutaron.

Para las pruebas de caja negra se elabora una tabla resumen, con el siguiente formato:

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa

Donde**:**

Elemento: especifica el nombre del elemento a probar (Documento o Aplicación).

No: un número que identifique al error o no conformidad.

No conformidad*:* se especifica el error detectado, la no conformidad encontrada.

Aspecto correspondiente: se describe la no conformidad, cualquier otro aspecto relevante y se especifica su localización.

Etapa de la detección: en qué etapa del ciclo de vida se detectó el error (Revisión de la Documentación o iteración en la Aplicación).

Significativa: se marca con una <X> si la no conformidad se considera significativa.

No Significativa: se marca con una <X> si la no conformidad se considera no significativa.

Dentro de los resultados de estas pruebas se encuentra la detección de 101 No Conformidades tanto en la documentación como en la aplicación, siendo el 68,3% de estas significativas.

3.1.2 Cobertura de los casos de pruebas

Con la puesta en práctica de los quince casos de prueba diseñados se detectaron errores en el sistema con siete de ellos.

El resultado de la cobertura de las pruebas realizadas es el siguiente:

- ✓ Casos de prueba realizados = 15 / 15= 100 %
- ✓ Casos de prueba que no arrojaron errores en el sistema: 8/15 = 53.3 %
- ✓ Casos de prueba que arrojaron errores en el sistema: 7/15 = 46.7 %

En estos 7 casos de prueba se obtuvo como resultado la detección de 13 No Conformidades que serán detalladas en el epígrafe 3.2.3.

3.2 Análisis de los defectos encontrados. Análisis de los resultados.

En algunos casos de pruebas cuando fueron ejecutados se detectaron varios errores en la aplicación y en la documentación, los que serán descritos en detalladamente en los epígrafes 3.2.1, 3.2.2 y 3.2.3.

Los resultados de las pruebas necesitan ser determinados siempre que ocurra la prueba y la ejecución de la evaluación. Esta última puede ocurrir muchas veces durante el ciclo de vida del desarrollo, los resultados de estas deben ser determinados y almacenados de una manera tal que puedan ser repasados y evaluados individualmente para cada caso de ejecución de prueba.

El sistema ha sido probado en su totalidad, los resultados obtenidos en las diferentes iteraciones han sido registrados y entregados al jefe de proyecto y a su vez estos han sido corregidos.

La última versión de las pruebas realizadas al sistema, no muestran prácticamente errores debido a que los fallos se han corregido según se han ido detectando.

Se concluye que con la planificación de las pruebas, con la información obtenida de los elementos de software a ser probados y de los requisitos, se obtuvo buenos resultados en cuanto a la calidad y eficiencia del producto, demostrado con el resultado de las pruebas.

3.2.1 No Conformidades

Una no conformidad es la falta de cumplimiento de especificaciones establecidas, o la producción bajo un procedimiento no aprobado o con algún error (ISO 9001:2005).

Aun cuando los procedimientos y controles buscan asegurar que los procesos permanezcan bajo control, es inevitable que a veces ocurran errores, los cuales son definidos como cualquier desajuste con respecto a los procedimientos operativos aprobados, que pueden afectar la calidad del producto o servicio. Con el propósito de manejar tales anomalías, un sistema de calidad debe incluir la gestión de No Conformidades. Una gestión adecuada no se limita a aplicar un "parche" para la resolución de reclamos, sino que identifica y elimina la verdadera causa de los errores.

En el caso del sistema alasLIPO se detectaron un total de 101 No Conformidades (Fig. 3.1), 88 en la etapa de la revisión de la documentación y 13 en la revisión de la aplicación a través de los diseños de casos de pruebas realizados. Se hicieron un total de 36 recomendaciones al equipo de desarrolladores.

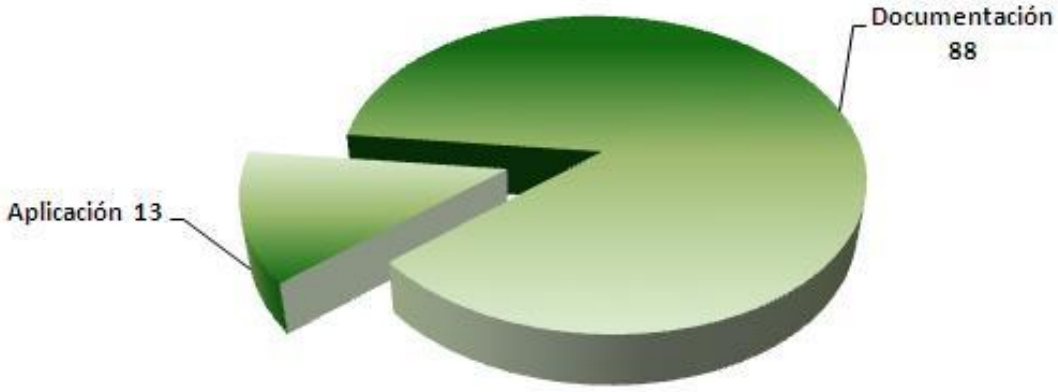

Fig. 3.1 Total de No Conformidades detectadas.

3.2.2 Liberación de la Documentación.

La tabla siguiente (Fig. 3.2) muestra como durante el proceso de revisión de la documentación se detectaron un total de 88 No Conformidades, 69 de ellas significativas y 19 no significativas, así mismo se hicieron un total de 36 recomendaciones importantes al grupo de desarrolladores del sistema.

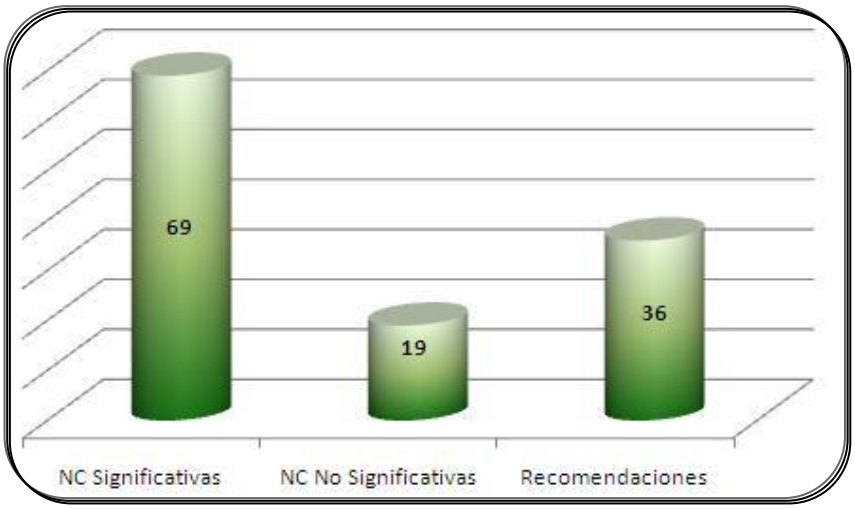

Fig. 3.2 No Conformidades y Recomendaciones en la Documentación.

A continuación se desglosa por iteraciones (Fig. 3.3) como se comportaron los indicadores de No Conformidades significativas y recomendaciones en el proceso de prueba, así mismo detallado por documentos, haciendo referencia en éstos a las No Conformidades significativa, no significativa y recomendaciones (Fig. 3.4 – 3.8).

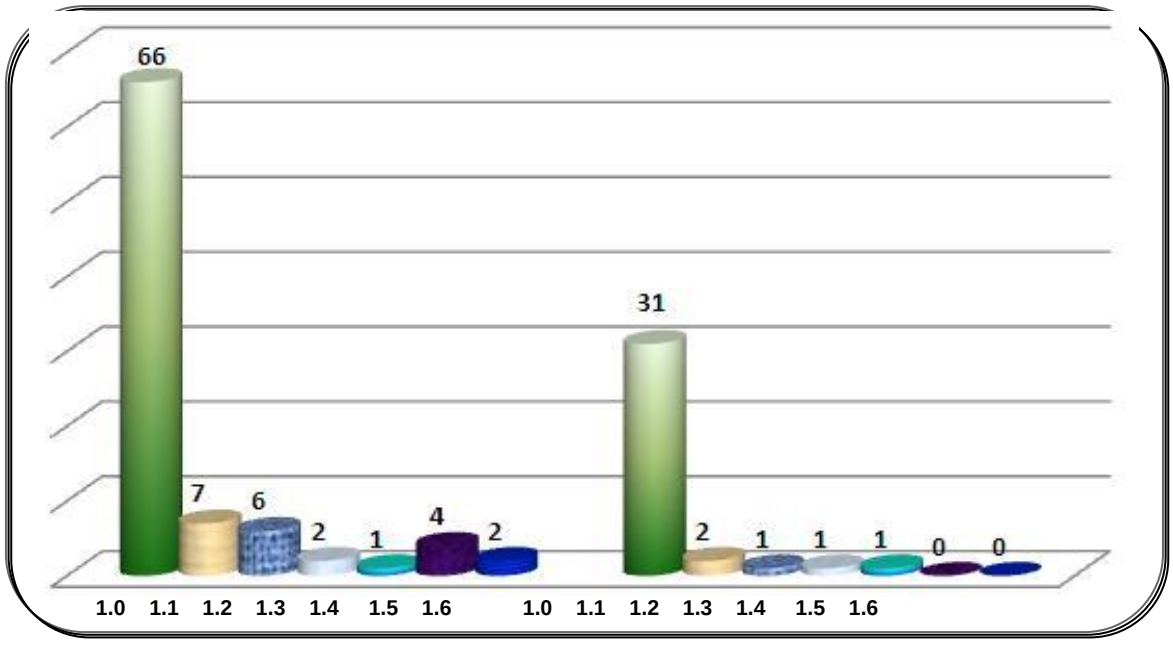

Fig. 3.3 No Conformidades y Recomendaciones por Iteraciones.

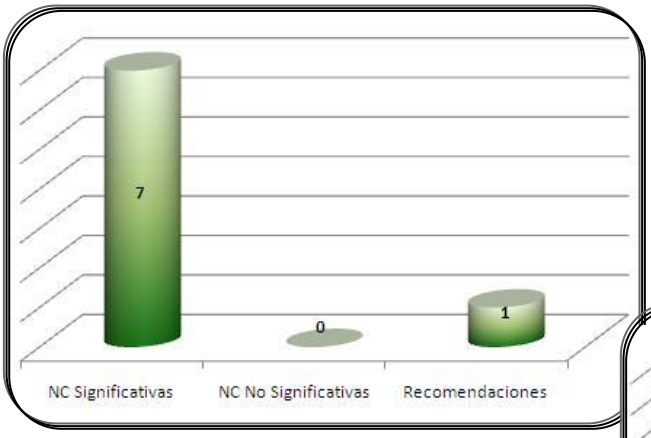

Fig. 3.4 No Conformidades y Recomendaciones Diccionario de Datos.

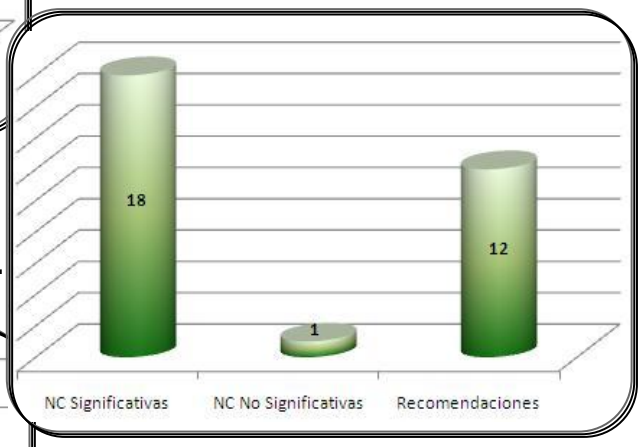

Fig. 3.5 No Conformidades y Recomendaciones. Especificación de Requisitos.

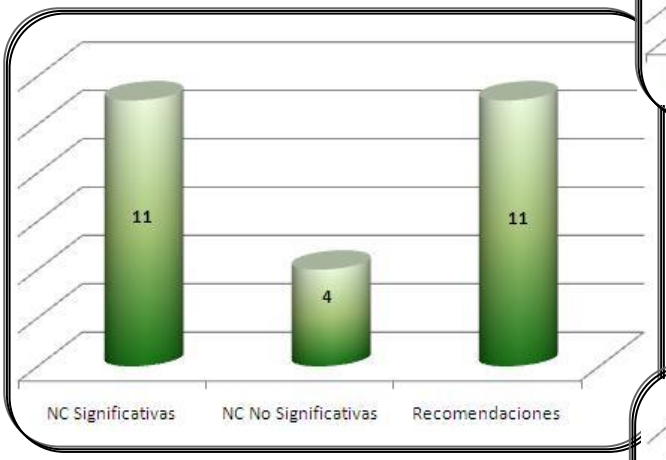

Fig. 3.6 No Conformidades y Recomendaciones. Manual de Usuario.

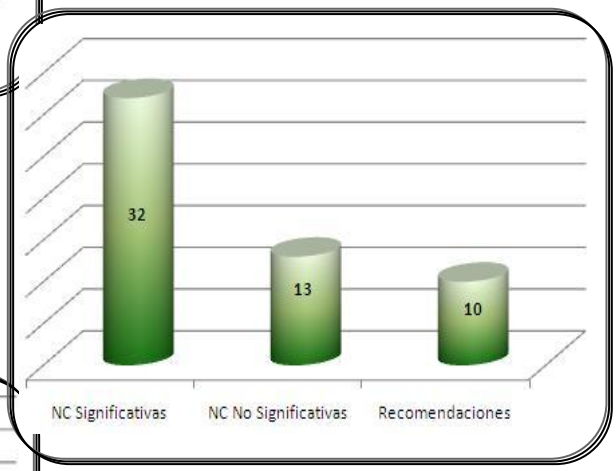

Fig. 3.7 No Conformidades y Recomendaciones. Modelo de Caso de Uso del Sistema.

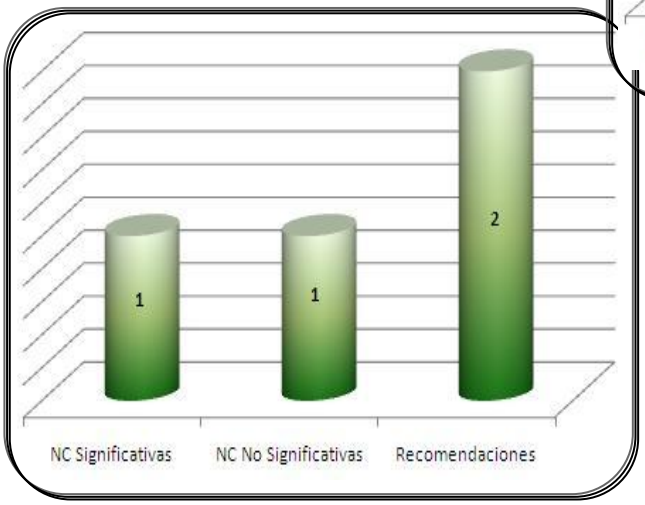

Fig. 3.8 No Conformidades y Recomendaciones. Roles y Permisos.

3.2.3 Liberación de la Aplicación.

La siguiente tabla (Fig. 3.9) resume la información derivada del diseño de los casos de prueba y aplicación de las pruebas pertinentes, mostrando las No Conformidades detectadas y las recomendaciones efectuadas en las dos iteraciones o versiones realizadas.

De los 15 casos de pruebas diseñados se obtuvo un total de 12 No Conformidades en la primera iteración o versión y 1 en la segunda, para un total de 13, vale aclarar que la totalidad de las encontradas fueron significativas, no se hizo ninguna recomendación al equipo de los desarrolladores al respecto.

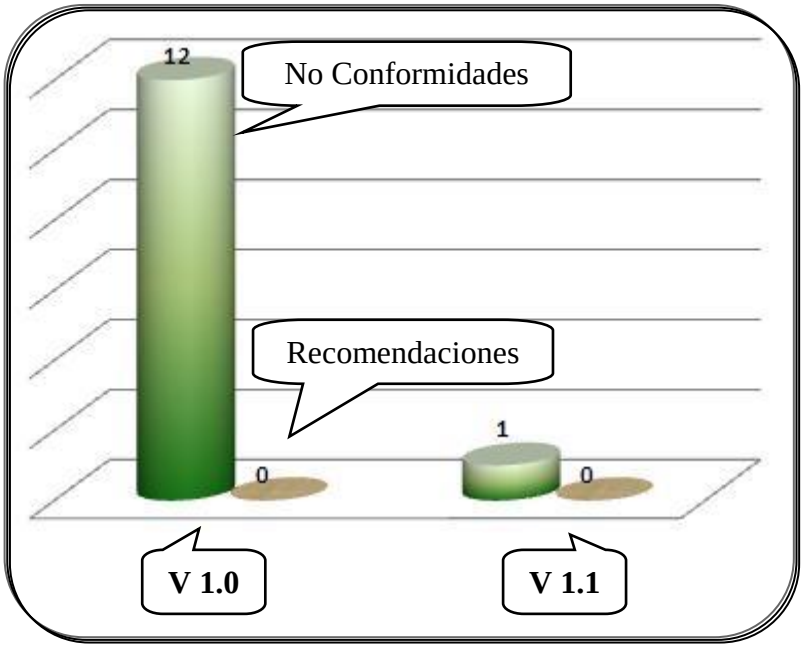

Fig. 3.9 Total de No Conformidades de la Aplicación.

3.2.4 Prueba de Carga y Estrés.

3.2.4.1 Entorno de Prueba.

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se le realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probar dicho producto se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso.

Por ello se tuvo en cuenta a la hora de llevar a cabo todo el proceso de pruebas al sistema alasLIPO, algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las pruebas, en aras de que las mismas lograran minimizar los errores en la aplicación, aunque vale aclarar que no son los óptimos que se necesitan para el mejor funcionamiento de la herramienta. A continuación se hace mención a algunos de los requerimientos que se consideran necesarios para las pruebas y seguidamente se mostrarán las condiciones reales en que fueron realizadas las pruebas:

Para el servidor de aplicación (condiciones óptimas):

- ✓ PC con Microprocesador Pentium IV a 3.00 GHz o Superior.
- ✓ 2 GB de RAM mínimo o Superior.
- ✓ 40 GB de Disco Duro o Superior.
- ✓ Conexión TCP/IP.
- ✓ Lenguaje de programación del lado del servidor PHP versión 5.1.6.
- ✓ Debe tener instalado el servidor web Apache 2.0.

Para el servidor de aplicación (condiciones reales):

- ✓ PC con Microprocesador Pentium IV a 2.00 GHz.
- ✓ 1 GB de RAM.
- ✓ 80 GB de Disco Duro.
- ✓ Conexión TCP/IP.
- ✓ Servidor web Apache 2.0.

Para automatizar las pruebas de Carga y Estrés:

- ✓ Herramienta JMeter 2.3.
- ✓ Maquina virtual de Java: Java Virtual Machine 1.3 o superior.
- ✓ Navegador: Internet Explorer o Mozilla.
- ✓ Sistema Operativo Windows XP Profesional o Linux (Ubuntu).

3.2.4.2 Análisis de los resultados obtenidos.

Las pruebas de carga y estrés se ejecutaron utilizando una PC con sistema operativo Microsoft Windows XP Profesional, Service Pack 3, con 1 GB de memoria RAM y microprocesador Pentium IV.

Para la realización de estas pruebas se instaló la aplicación web en la PC, utilizando esta última como servidor al mismo tiempo, se instaló la máquina virtual de Java (JAVA SE RUNTIME ENVIRONMENT

(JRE) VERSION 6) y se ejecutó la herramienta JMeter. Se probaron cada uno de los siete Casos de Uso arquitectónicamente significativos, que tributan a los siete Diseños de Caso de Prueba probados. Las pruebas fueron realizadas por dos probadores que no pertenecen al equipo de desarrollo del sistema. Los resultados comprenden una serie de variables obtenidas como consecuencia de la ejecución de estas pruebas a los casos definidos anteriormente y al sistema como un todo.

Se realizaron tres pruebas con la simulación de 200 usuarios interactuando con el sistema desde la misma URL y con un período de subida de 1 segundo, bajo diferentes condiciones, que a continuación se relacionan:

1. 50 usuarios en 4 bucles, o sea, 50 conexiones en un primer momento, luego las próximas 50 y así hasta llegar a la totalidad de las 200.
2. 100 usuarios en 2 bucles. o sea, 100 conexiones en un primer momento y luego las próximas 100, llegando así a la totalidad de las 200.
3. 200 usuarios en 1 bucle, o sea, 200 conexiones desde un primer momento.

Los resultados totales de las pruebas se pueden observar en las imágenes que exporta la misma herramienta del Informe Agregado donde se relacionan todas las variables con sus respectivos resultados. (Ver Anexo 3, 4, 5).

A continuación, en la siguiente tabla se relacionan algunos parámetros que se creen válidos hacer referencia de forma general.

No	Variable	(50 en 4 bucles)	(100 en 2 bucles)	(200 en 1 bucle)
1	Muestras	200	200	200
2	Peticiones HTTP	329	357	292
3	Mediana	484	766	453
4	Media	2177	5366	5814
5	Max	12922	50703	48656
6	Línea 90%:	9000	29094	26860
7	Kb/Sec	17998,4	18487,3	18861,3
8	Rendimiento	1,9/sec	2,1/sec	2,0/sec

- ✓ **Muestras:** Número de muestras para cada URL.
- ✓ **Peticiones HTTP**: Cantidad de peticiones HTTP realizadas al servidor.
- ✓ **Mediana:** Tiempo promedio que han tardado en cargarse las páginas.
- ✓ **Media:** Media del tiempo total que demoraron las peticiones en cargarse.
- ✓ **Max:** El máximo tiempo transcurrido para las muestras de la URL dada.
- ✓ **Línea 90%:** Tiempo máximo en que corrieron el 90 % de las peticiones reales.
- ✓ **Kb/Sec:** Velocidad de carga de las páginas.
- ✓ **Rendimiento:** Representa el número de muestras por unidad de tiempo.

Según se refleja en la tabla anterior la aplicación toma un comportamiento diferente para todas las simulaciones realizadas pero que están relativamente comprendidas en diferentes rangos según el atributo que se relacione.

Cuando se la cantidad de usuarios conectados en un segundo llega a la máxima cantidad prevista disminuye la cantidad de peticiones HTTP realizadas con respecto a cuándo se encuentran la mínima cantidad prevista.

Se puede decir además que según va aumentando la cantidad de usuarios simulados en el mismo instante de tiempo la velocidad de carga de las páginas va en ascenso, de igual forma se comporta la media del tiempo total que demoran las peticiones en cargarse.
Por su parte el rendimiento del sistema se mantiene relativamente contante sobre las 2 unidades por segundos.

Analizando los resultados arrojados por la herramienta podemos decir que la aplicación alasLIPO soporta una gran carga y estrés bajo condiciones anormales de concurrencia de usuarios y de flujo de datos, manteniendo los parámetros relativamente estables en todos los casos simulados con una velocidad de carga aceptable y un rendimiento estable. El tiempo máximo que puede esperar un cliente en que se carguen las diferentes páginas del sistema es de 15 segundos, tiempo muy reconfortante para el usuario durante su navegación por el sistema.

3.3 Evaluación del Producto según los parámetros de calidad.

En la elaboración de cualquier producto la calidad es uno de los aspectos más importantes para determinar si éste es mejor o peor que otro, y el desarrollo de productos interactivos no puede prescindir de este factor, sino más bien todo lo contrario [26].

Para la evaluación del producto de acuerdo a los parámetros de calidad se aplicó una lista de chequeo (Anexo 6) al sistema con el apoyo del equipo de desarrollo del mismo, se obtuvieron una serie de resultados que permitieron dar una valoración por cada atributo para al final dar una evaluación general del sistema.

Para evaluar los atributos de calidad, se utilizó una lista de chequeo, definiendo un conjunto de preguntas que tienen un valor máximo de 5 puntos. A cada una de ellas se le otorgó una calificación de 0 a 5 puntos, en función del grado de satisfacción o de cumplimiento en el sistema, Se procedió luego a promediar la puntuación obtenida para cada atributo y se emitió una evaluación, a partir de los diferentes rangos que se definen a continuación:

- ✓ **((0 – 2): Atributo no disponible.**
- ✓ **(3 – 4): Atributo parcialmente disponible.**
- ✓ **(4 – 5): Atributo disponible**

3.3.1 Parámetros de Calidad. Seguridad.

Con esta prueba se busca verificar que los mecanismos de protección incorporados en el sistema realmente lo protegerán de accesos impropios.

Se persigue validar que en la aplicación:

- ✓ Los datos del sistema solo pueden ser accesibles por los autores debidamente autorizados.
- ✓ Las funciones del sistema solo pueden ser accesibles por los autores debidamente autorizados.
- ✓ Las funciones que atenten contra la integridad de los datos de negocios sean debidamente impedidas.

En alasLIPO se definieron para este atributo 4 preguntas. Luego de su aplicación se obtuvo la máxima puntuación, que al promediar arrojó del mismo modo un valor de 5 puntos, ubicándose en el rango superior, llegando a la conclusión que es una aplicación en la que el parámetro de la Seguridad fue muy bien definido, estructurado y aplicado finalmente. Por lo tanto es una **aplicación segura.** Lo antes explicado se detalla en la siguiente tabla.

Total de Preguntas	Evaluadas de:						Promedio	Evaluación según rango.
	0	1	2	3	4	5		
4	-	-	-	-	-	4	5	Atributo Disponible

3.3.2 Parámetros de Calidad. Portabilidad.

La portabilidad es uno de los conceptos claves en la programación de alto nivel. Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del software es capaz de reutilizase en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

El prerrequisito para la portabilidad es la abstracción generalizada entre la aplicación lógica y las interfaces del sistema. Cuando un software se puede compilar en diversas plataformas se dice que es multiplataforma. Esta característica es importante para el desarrollo de reducción de costos, cuando se quiere hacer una misma aplicación.

En algunos casos el software es "independiente" de la plataforma y puede ejecutarse en plataformas diversas sin necesidad de ser compilado específicamente para cada una de ellas, a este tipo de software se le llama interpretado, porque necesita un intérprete para ser ejecutado en las diferentes plataformas.

En alasLIPO se definieron para este atributo 2 preguntas. Luego de su aplicación se obtuvo el 100% de los puntos, al calcular su promedio para definir el rango y poder dar una evaluación se obtuvo el valor de 5 puntos, encontrándose éste en un rango máximo, llegando a la conclusión de que este atributo está totalmente disponible, por lo que se considera que es una **aplicación portable.** Lo antes explicado se detalla en la siguiente tabla.

Total de Preguntas	**Evaluadas de:**						**Promedio**	**Evaluación según rango.**
	0	**1**	**2**	**3**	**4**	**5**		
2	**-**	**-**	**-**	**-**	**-**	**2**	**5**	**Atributo Disponible**

3.3.3 Parámetros de Calidad. Usabilidad.

Uno de los parámetros de calidad más determinantes con relación a los sistemas interactivos es la Usabilidad de los mismos [28], hasta el punto que las más prestigiosas estandarizaciones (ISO 91, IEEE 98) vienen considerando este factor desde hace mucho tiempo.

Muchas son las ventajas que la usabilidad puede proporcionar y por ello debería ser tratada como un factor de calidad estratégico y relevante [29].

Existen muchos métodos de evaluación de la usabilidad que se pueden clasificar de diversas maneras.

Evaluación automática.

Consiste en el uso de software que detecta problemas elementales, como por ejemplo:

- ✓ Tamaños absolutos de fuentes y de tablas.
- ✓ Formato de los textos.
- ✓ Tamaño de las páginas.
- ✓ Tiempos de descarga.
- ✓ Enlaces rotos.

Evaluación de acuerdo a directrices.

Otra forma de evaluación consiste en considerar la adecuación de las características del sitio a alguna(s) lista(s) de directrices o características que debe tener un "buen sitio". Sólo tiene valor real si lo realizan como mínimo dos personas que ya tengan cierta experiencia, y se obtiene una lista de cuestiones con las que se puede realizar un rediseño alternativo del sitio.

La lista más conocida es la de los 10 heurísticos de Nielsen, enumerados aquí a título ilustrativo:

- ✓ Visibilidad del estado del sistema.
- ✓ Emparejamiento entre el sistema y el mundo real.
- ✓ Control y libertad del usuario.
- ✓ Consistencia y estándares.
- ✓ Prevención de errores.
- ✓ Reconocimiento sobre recuerdo.
- ✓ Flexibilidad y eficiencia de uso.
- ✓ Estética y diseño minimalista.
- ✓ Ayuda a reconocer, diagnosticar y solucionar errores.
- ✓ Ayuda y documentación.

En alasLIPO se definieron para este atributo 29 preguntas. Luego de su aplicación no se obtuvo la máxima puntuación, al promediar arrojó un valor de 4,4 puntos, manteniéndose en el rango máximo, llegando a la conclusión de que este atributo está disponible, aunque no en un 100%, por lo que se considera que es una **aplicación usable**. Lo antes explicado se detalla en la siguiente tabla.

Total de Preguntas	Evaluadas de:						Promedio	Evaluación según rango.
	0	1	2	3	4	5		
29	7	-	-	-	2	20	4.4	Atributo Disponible

En el caso de este atributo las preguntas que no obtuvieron el máximo de puntuación se relacionan a continuación:

- 0 puntos.
  - ✓ ¿Se consideran otros idiomas para las instrucciones en la pantalla de forma adecuada?
  - ✓ ¿Hay ausencia de términos en idiomas diferentes mezclados?
  - ✓ ¿Se posibilita la opción de incrementar el tamaño de los caracteres?
  - ✓ ¿Permite al usuario interrumpir su tarea y continuar más tarde?
  - ✓ ¿Permite deshacer las acciones, e informar el estado?
  - ✓ ¿Proporciona funciones deshacer, rehacer?
  - ✓ ¿Existen diferentes niveles de ayuda?

- 4 puntos.
  - ✓ ¿Se permite la utilización del ratón o el teclado?
  - ✓ ¿Se reconoce todas las tareas del software en cualquier momento?

3.3.4 Parámetros de Calidad. Confiabilidad.

La confiabilidad en un sistema está dada por la capacidad de la recuperación del software ante fallas. Los datos almacenados no pueden ser corrompidos fácilmente por códigos defectuosos, accesos simultáneos, o finalización inesperada de los procesos.

Confiabilidad-Consistencia sobre carga

Todo sistema tiene límites de capacidad. ¿Qué pasa cuando estos límites son excedidos? El sistema no debería jamás perder o corromper la información.

Confiabilidad-Consistencia bajo concurrencia

Los sistemas que permiten accesos simultáneos por usuarios múltiples, o los que usan concurrencia interna deberían estar libres de condiciones de carrera y bloqueo.

Confiabilidad-Disponibilidad bajo carga

Todo sistema tiene límites de capacidad. ¿Qué pasa cuando estos límites son excedidos? El sistema debería continuar sirviendo aquellas solicitudes que es capaz de manejar. No debería interrumpirse o detener el proceso de todas las solicitudes.

Confiabilidad-Longevidad

El sistema debería continuar operando tanto como lo necesite. No debería gradualmente terminarse los recursos disponibles. Ejemplos de defectos de longevidad incluyen fugas de memoria o el agotamiento de espacio en discos con archivos de registro.

En alasLIPO se definieron para este atributo 2 preguntas. Luego de su aplicación se obtuvo la máxima puntuación, que al promediar arrojó del mismo modo un valor de 5 puntos, ubicándose en el rango superior, llegando a la conclusión que es una aplicación en la que el parámetro de la Confiabilidad fue muy bien definido y estructurado. Por lo tanto es una **aplicación confiable**. Lo antes explicado se detalla en la siguiente tabla.

Total de Preguntas	Evaluadas de:						Promedio	Evaluación según rango.
	0	1	2	3	4	5		
2	-	-	-	-	-	2	5	Atributo Disponible

Haciendo un análisis de los resultados expuestos anteriormente (Ver Fig. 3.10) podemos llegar a las siguientes conclusiones:

- ✓ El software alasLIPO es una aplicación segura.
- ✓ El software alasLIPO es una aplicación portable.
- ✓ El software alasLIPO es una aplicación usable.
- ✓ El software alasLIPO es una aplicación confiable.

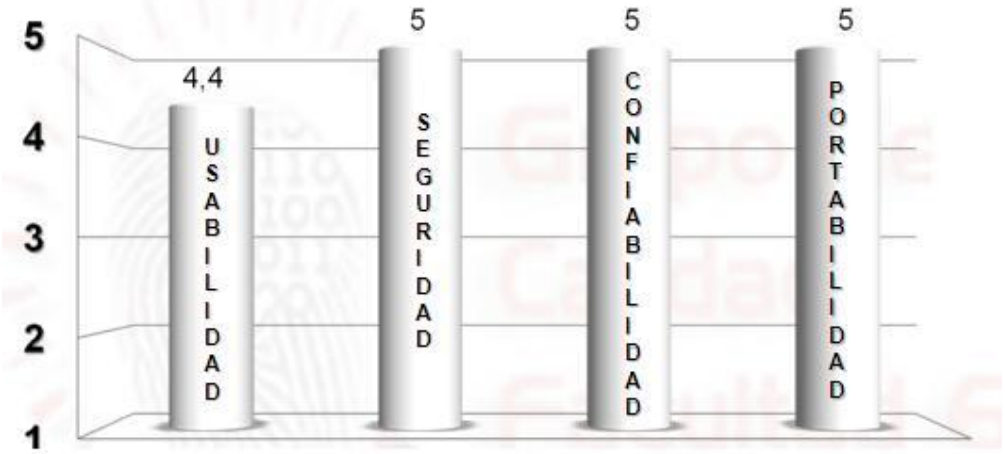

Fig. 3.10 Resultados de las Listas de Chequeo de los Parámetros de Calidad.

Conclusiones

Luego de haber concluido esta etapa del proceso de liberación se obtuvo una serie de resultados concretos de las pruebas realizadas.

Con la puesta en práctica de las pruebas exploratorias y de Caja Negra, a través de la técnica de partición de equivalencia se detectaron un total de 101 No Conformidades, 88 en la revisión de la documentación y 13 en la aplicación, haciendo un total de 36 recomendaciones al grupo de desarrolladores.

Además se aplicaron las Pruebas de Carga y Estrés con la herramienta Apache JMeter, obteniéndose una serie de resultados definidos por cada una de las variables que arroja la herramienta donde se pudo llegar a la conclusión de que alasLIPO soporta una gran carga y estrés bajo condiciones anormales de concurrencia de usuarios y de flujo de datos, manteniendo los parámetros relativamente estables en todos los casos simulados con una velocidad de carga aceptable y un rendimiento estable. El tiempo máximo que puede esperar un cliente en que se carguen las diferentes páginas del sistema es de 15 segundos, tiempo muy reconfortante para el usuario durante su navegación por el sistema

Por último se aplicó una lista de chequeo con diferentes parámetros de calidad concluyendo finalmente que el sistema en cuestión es seguro, usable, confiable y portable.

Conclusiones Generales

Con la realización de este trabajo se logró desarrollar las pruebas de liberación de la documentación y de la aplicación alasLIPO. Para ello se realizó una revisión técnica de la documentación, se diseñaron y ejecutaron un conjunto de pruebas que permitieron obtener un software con un número reducido de defectos, emitiendo una evaluación al finalizar el proceso de prueba definido.

Con este trabajo se arribaron además a las siguientes conclusiones:

- ✓ La revisión técnica de la documentación permitió evaluar tanto la ortografía y redacción, como la calidad de los artefactos generados durante la concepción y desarrollo de la aplicación, garantizando, luego de las correcciones pertinentes, un mejor estado de estos y su posterior liberación.

- ✓ La aplicación de las Pruebas de Caja Negra diseñadas permitió detectar en el proceso de liberación los elementos que se omitieron durante la implementación o no se encontraban correctamente implementados según lo descrito en la documentación, necesarios para dar cumplimiento de forma óptima a los requisitos funcionales establecidos por el cliente.

- ✓ Con la ejecución de las pruebas de Carga y Estrés, se pudo evaluar de satisfactorio el comportamiento y rendimiento del sistema funcionando como un todo y bajo condiciones extremas.

- ✓ La realización de las pruebas de liberación al sistema alasLIPO garantizó la obtención de un software con un número reducido de defectos, permitiendo que todas las funcionalidades que se especificaron al inicio por parte de los clientes fueran cumplidas satisfactoriamente.

Recomendaciones.

Luego de haber concluido con todo el proceso de prueba y dada la importancia de la detección de errores a tiempo en las aplicaciones Web, a todas aquellas personas o empresas productoras de software, se recomienda:

- ✓ Realizar todos los esfuerzos posibles por instituir correctos diseños de pruebas con el fin de corregir todos los posibles defectos del sistema.

- ✓ Adaptar estas pruebas a nuevos productos de la misma línea para lograr un estándar de pruebas que demuestren la usabilidad de los sistemas.

- ✓ Realizar otros tipos de pruebas al sistema, como son las pruebas de Caja Blanca.

- ✓ Utilizar los conceptos, ejemplos y resultados de este trabajo para lograr que los grupos de desarrollo conozcan las técnicas de prueba y que las incluyan como una fase más dentro del desarrollo de los sistemas.

- ✓ Que el documento obtenido como resultado de esta investigación quede como fuente de consulta y bibliografía en este tema de pruebas de software.

Referencias Bibliográficas

1. R.B Laitinen, Mauri, Fayad, Mohamed and Ward, Robert. *Software Engineering in the Small. Comunications of the ACM.* 2000.
2. Sastre, Benjamín del. Portal Universi Argentina. [En línea] 2006.
3. Febles, Ailin. *Modelo de Referencia para la Gestión de Configuración en la pequeña y mediana empresa de software.* Ciudad de la Habana: s.n., 2001. Tesis doctoral.
4. PAZ, A. C. *El modelo de mccall como aplicación de la calidad a la revisión del software de gestión empresarial.1997.*
5. PRESSMAN, R, S. Ingeniería del Software. Un enfoque práctico. Quinta edición. ed. McGraw Hill, 2005. vol. I, 500 p.
6. SALANOVA, P. E. (2006). Modelos de Calidad Web. Clasificación de Métricas., UNIVERSIDAD NACIONAL DE EDUCACION A DISTANCIA: 308.
7. GUTIÉRREZ, J. J.; ESCALONA, M. J., et al. *Estudio comparativo de propuestas para la generación de casos de prueba a partir de requisitos funcionales.* Disponible en: http://www.lsi.us.es/docs/informes/LSI-2005-01.pdf
8. Ivar Jacobson, G.B., James Rumbaugh, El proceso unificado de desarrollo de software. 2000, Madrid.
9. Conferencia # 5. Pruebas. Ingeniería del Software II. Curso 2007 – 2008.
10. Artículo "Fundamentos para las Pruebas de Software". [Julio-Agosto de 2005] http://www.e-quallity.com.mx/articulos/SG-200504-Luis04.pdf
11. Pressman, Roger S. Ingeniería del Software, Un enfoque práctico. 2005
12. Articulo "Ciclo de vida del Software" [Febrero de 2008] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php
13. Pressman, Roger S. Ingeniería del Software, Un enfoque práctico. 2005.
14. Jakarta, jakarta.apache.org/jmeter/.
15. Senn, James. Pruebas de rendimiento del software. Mcgraw Hill 1997.
16. Fernández Vilas, Ana. "Diagrama de Casos de Uso". http://tvdi.det.uvigo.es/~avilas/UML/node25.html
17. Documento de Arquitectura del Software (SICAL). Versión 2.0. http://svn2.assembla.com/svn/SICAL/trunk/documentacion/Gestion%20del%20proyecto/Documento%20de%20Arquitectura%20de%20Software%20-%20SAD.doc

18. Jiménez Darwin, Eduardo Aguirre Carlos. Universidad Antonio Nariño. Presentación de Pruebas. http://www.slideshare.net/dajigar/presentacion-pruebas-presentation
19. Compuware, QALoad. 2007.
20. Compuware, Compuware. 2006.
21. Mercuryinteractive.Com, LoadRunner.
22. Noticias.Com, www.noticias.com/noticia .
23. BamSol, www.bamsol.com/bamsol 2008.
24. Pérez, B; Pittier, A; Travieso M; Wodzislawski M. "Testing exploratorio en la práctica."; http://www.ces.com.uy/documentos/JIISIC-2007.pdf
25. Bach, J; "Exploratory Testing Explained". The Test Practicioner. 2002; http://www.satisfice.com/articles/et-article.pdf
26. Robinson, H; "Exploratory Modeling"; http://www.testingcraft.com/exploratory-robinson.html
27. *Caso de prueba*. Enero 2007 [cited; Available from: http:// lsi.ugr.es/~arroyo/inndoc/doc/Caso_de_prueba
28. Bevan N., "Quality in Use: Meeting User Needs for Quality", Journal of System and Software (1999).
29. Constantine, L.L., Loockwood L.A.D., "Software for Use: A Practical guide to the Models and Methods of Usage-Centered Design". Addison-Wesley (1999).

Bibliografía

1. *Juan Manuel. Cueva,* http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

2. Un enfoque actual sobre al calidad del software, *Oscar M. Fernández Carrasco, Delba García León y Alfa Beltrán Benavides,* http://bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm.

3. Calidad en ingeniería del software, dmi.uib.es/~bbuades/calidad/index.htm

4. De Domingo, J. y Arranz, A., Calidad y mejora continua. Ed Donostiarra. 2007.

5. Gestión de la calidad del software, http://www-
306. ibm.com/software/info/ecatalog/es_ES/rational/SW730.html

6. Modelos de calidad de software y software libre, *Ernesto Quiñones A.*,
http://www.eqsoft.net/presentas/modelos_de_calidad_y_software_libre.pdf

7. Control de la calidad del software, http://www.sdl.com/es/services/services-sdl-languageservices/services-sdl-software-qa.htm.

8. Pruebas de software, http://lsi.ugr.es/~ig1/docis/pruso.pdf

9. Pruebas de software terminado, http://gbtcr.chileforge.cl/info_web/node139.html

10. Aplicación práctica del diseño de pruebas de software a nivel de programación,
http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf

14. Pruebas de software, http://lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt

15. Pruebas de software,
http://www.greensqa.com/archivos/Servicio%20Pruebas%20de%20Software.pdf

16. Pruebas del software, http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf

17. http://www.seccperu.org/?q=node/389

18. Bedini G, Alejandro. *Calidad Tradicional y de Software.*

19. Pressman, Roger S. *Ingeniería del software. Un enfoque práctico.* Cuarta. s.l. : McGraw Hill, 1998.

20. Laitinen, Mauri, Fayad, Mohamed and Ward, Robert. *Software Engineering in the Small.Comunications of the ACM.* 2000.

21. Sastre, Benjamín del. Portal Universi Argentina. [En línea] 2006.

22. Febles, Ailin. *Modelo de Referencia para la Gestión de Configuración en la pequeña y mediana empresa de software.* Ciudad de la Habana: s.n., 2001. Tesis doctoral.

23. Hall, Tracy, Rainer, Austen and Baddoo, Nathan. *Implementing Software Process Improvement: An Empirical Study. Software Process: Improvement and Practice.* 2002.

24. Kulpa, Margaret and Johnson, Kent. *Interpreting the CMMI: A Process Improvement Approach.*2003.

25. McFeeley, Robert. *IDEAL: A Users Guide for Software Process Improvement, Handbook.* Pittsburgh: Software Engineering Institute, Carnegei Mellon University, 1996.

26. Di Mónaco, Silvia y Vitali, Silvio. *El Modelo IDEAL para implementar CMMI.* [PowerPoint] s.l.: Centro de Calidad en Tecnologías de la información, 2005.

27. Calvo Manzano, J.A. Experiences in the Application of Software Process Improvement in SMES. s.l.: Software Quality Journal, 2002. págs. 261-273.

28. Straub, Pablo. *El costo de la calidad.* 2006.

29. *Estimación del coste de la calidad del software a través de la simulación del proceso de desarrollo.* Ruiz, Mercedes y Ramos, Isabel. 1, Revista Colombiana de computación, Vol. II, págs. 75-87.

30. Guzmán, Adolfo. *Medición de la calidad del Software.* s.l. : Centro de Investigación en Computación.

31. Jacobson, I., Booch, G. y Rumbaugh, J. *Proceso Unificado de Desarrollo de Software.* Vol. I.

32. Patricio, Letelier. *Pruebas del Software.* Valencia: Departamento de Sistemas Informáticos y Computación.

33. Nielsen J. Usability Engineering. (1993). AP Professional. 1993. Massachusetts, USA.

34. Conferencia # 5. Pruebas. Ingeniería del Software II. Curso 2007 – 2008

Anexos

ANEXO 1. Diseño de Casos de Prueba

<Nombre del Proyecto>

Módulo <Nombre del Módulo>

Versión del proyecto

Diseño de Casos de Prueba

Nombre del Caso: <Nombre del Caso de Uso>

Versión del CP

Descripción General

[Descripción general del CU]

Condiciones de Ejecución:

[Precondiciones del CU].

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

Descripción de variable.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción

SC 1: <Sección #1 a revisar>

Id del escenario	Escenario	Variable 1 *(Nombre de la variable)*	Variable 2 *(Nombre de la variable)*	Variable N *(Nombre de la variable)*	Respuesta del Sistema	Resultado de la Prueba

Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta d Equipo Desarrollo

ANEXO 2. No Conformidades Detectada

<Nombre del Proyecto>

Módulo <Nombre del Módulo>

Versión 1.1

No Conformidades Detectada

Elemento de Configuración: <Nombre del Elemento>

Versión 1.1

Aspectos generales

Elementos Probados.

Elementos no Probados y causas.

Tabla de No Conformidades Detectadas

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa

Recomendaciones

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección

ANEXO 3. Resultados de las Pruebas en el JMeter. Informe Agregado.

50 hilos en 4 bucles.

Informe Agregado

Nombre: Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/	50	9824	9657	12281	4625	12922	0,00%	1,9/sec	12099,5
/js/SpryAccordion.js	50	289	375	547	0	1000	0,00%	3,8/sec	57251,4
/js/Fecha.js	50	495	391	1156	0	1546	0,00%	3,0/sec	1404,8
/css/sfcss.css	48	554	375	1093	0	4094	0,00%	28,2/min	3667,8
/css/main.css	37	1097	468	3640	0	5125	0,00%	26,1/min	6576,7
/css/SpryAccordion....	26	1528	750	4015	0	9000	0,00%	11,3/min	766,1
/images/logo.png	10	1810	532	7078	16	7078	0,00%	3,9/min	1954,8
/images/bannerbg.j...	7	229	359	484	0	484	0,00%	22,2/min	13976,0
/images/alimentos....	7	611	437	2609	0	2609	0,00%	9,5/min	1682,4
/images/hipertensi...	7	1370	1469	3234	0	3234	0,00%	3,3/min	374,4
/images/conocete.gif	6	1401	547	5453	0	5453	0,00%	2,5/min	98,3
/images/hacerdieta...	5	1650	500	5766	375	5766	0,00%	1,9/min	46,4
/images/bottom.gif	4	1324	1484	3313	16	3313	0,00%	1,5/min	5,4
/images/accept.png	3	318	375	547	32	547	0,00%	1,1/min	14,7
/images/symfony.gif	3	484	390	1047	16	1047	0,00%	1,1/min	30,7
/images/banner_m...	3	463	500	531	360	531	0,00%	1,1/min	56,1
/sfGuardAuth/regist...	4	3398	3187	6015	1203	6015	0,00%	1,5/min	236,5
/sf/prototype/js/prot...	2	789	1031	1031	547	1031	0,00%	58,2/min	69118,3
/images/arrow.gif	2	195	375	375	15	375	0,00%	2,7/sec	160,0
/images/indicator.gif	2	1351	1546	1546	1156	1546	0,00%	38,8/min	1428,8
/images/cancel.png	2	367	375	375	359	375	0,00%	1,8/sec	1538,3
/herramientas/mnp	1	782	782	782	782	782	0,00%	38,4/min	2795,4
TOTAL	329	2177	484	9000	0	12922	0,00%	1,9/sec	17998,4

ANEXO 4. Resultados de las Pruebas en el JMeter. Informe Agregado.

100 hilos en 2 bucles.

Informe Agregado

Nombre: Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

Label	# Muestras	Media	Mediana	Linea de 90...	Mín	Máx	% Error	Rendimiento	Kb/sec
/	53	26079	30562	31547	390	32000	0,00%	19,7/min	2075,5
/js/SpryAcc...	47	711	765	1235	0	1656	0,00%	11,3/sec	168752,2
/js/Fecha.js	47	439	406	797	15	1219	0,00%	9,6/sec	4579,5
/css/sfcss....	47	608	469	860	0	1172	0,00%	6,0/sec	46808,3
/css/main.c...	47	877	781	1172	15	5031	0,00%	3,1/sec	46700,1
/css/SpryAc...	37	6305	766	20953	391	50703	0,00%	18,9/min	1274,5
/images/lo...	11	11680	8734	22828	391	43078	0,00%	5,7/min	2897,1
/images/ba...	3	3526	766	9438	375	9438	0,00%	9,5/min	6009,1
/images/ali...	3	625	375	1125	375	1125	0,00%	1,3/sec	14180,0
/images/hi...	3	276	375	438	16	438	0,00%	1,9/sec	12607,5
/images/co...	3	396	375	438	375	438	0,00%	2,5/sec	6073,2
/images/ha...	3	260	375	390	15	390	0,00%	2,6/sec	3750,6
/images/bo...	3	250	375	375	0	375	0,00%	2,7/sec	573,3
/images/ac...	3	250	375	375	0	375	0,00%	4,0/sec	3124,0
/images/sy...	3	542	422	782	422	782	0,00%	1,9/sec	3126,6
/images/ba...	3	0	0	0	0	0	0,00%	8,3/sec	24725,0
/sfGuardAu...	4	1793	2547	2547	906	2547	0,00%	29,2/min	4613,1
/sf/prototyp...	1	1281	1281	1281	1281	1281	0,00%	23,4/min	27814,6
/images/arr...	1	719	719	719	719	719	0,00%	41,7/min	41,7
/images/in...	1	344	344	344	344	344	0,00%	1,5/sec	3210,8
/images/ca...	1	1047	1047	1047	1047	1047	0,00%	28,7/min	407,4
/herramient...	1	1563	1563	1563	1563	1563	0,00%	19,2/min	1398,6
/login	4	949	984	1188	797	1188	0,00%	15,8/min	1667,2
/logout	2	781	782	782	781	782	0,00%	11,2/min	1184,7
/sfGuardUs...	4	406	406	407	406	407	0,00%	33,8/min	3557,2
/images/eli...	1	360	360	360	360	360	0,00%	1,4/sec	431,9
/images/ed...	1	359	359	359	359	359	0,00%	1,4/sec	628,1
/sfGuardUs...	3	411	407	422	406	422	0,00%	56,7/min	5977,3
/sfGuardUs...	2	594	766	766	422	766	0,00%	50,5/min	5323,8
/images/ok...	1	343	343	343	343	343	0,00%	1,5/sec	1336,7
/sfGuardUs...	1	406	406	406	406	406	0,00%	1,2/sec	7785,7
/sfGuardUs...	1	407	407	407	407	407	0,00%	1,2/sec	7766,6
/images/err...	1	359	359	359	359	359	0,00%	1,4/sec	1026,5
/sfGuardUs...	1	406	406	406	406	406	0,00%	1,2/sec	7785,7
/reporte/ge...	2	406	406	406	406	406	0,00%	20,1/min	2113,0
/js/validado...	1	359	359	359	359	359	0,00%	1,4/sec	3169,9
/images/bu...	1	360	360	360	360	360	0,00%	1,4/sec	161,1
/images/bu...	1	359	359	359	359	359	0,00%	1,4/sec	93,3
/reporte/rep...	1	766	766	766	766	766	0,00%	39,2/min	4126,6
/images/im...	1	719	719	719	719	719	0,00%	41,7/min	560,5
/images/xls...	1	359	359	359	359	359	0,00%	1,4/sec	1752,1
/imprimir/re...	1	781	781	781	781	781	0,00%	38,4/min	4047,4
/reporte_ex...	1	703	703	703	703	703	0,00%	42,7/min	2913,2
TOTAL	357	5366	766	29094	0	50703	0,00%	2,1/sec	18487,3

ANEXO 5. Resultados de las Pruebas en el JMeter. Informe Agregado.

200 hilos en 1 bucle.

Informe Agregado

Nombre: Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/	48	25772	26906	27875	8313	28250	0,00%	50,9/min	5364,9
/js/SpryAcco...	46	460	407	859	0	1250	0,00%	10,6/sec	158013,8
/js/Fecha.js	46	442	422	844	0	875	0,00%	5,7/sec	2720,7
/css/sfcss.c...	43	2199	391	828	0	39859	0,00%	21,1/min	2747,4
/css/main.c...	38	1716	406	1500	0	27031	0,00%	20,8/min	5245,3
/css/SpryAc...	30	5578	437	24515	0	48656	0,00%	16,7/min	1129,2
/images/log...	18	1789	781	6890	16	10469	0,00%	10,5/min	5316,0
/images/ba...	4	659	765	1484	31	1484	0,00%	2,6/min	1632,2
/images/ali...	3	1469	1078	3282	47	3282	0,00%	1,9/min	328,3
/images/hip...	3	2745	2875	5328	32	5328	0,00%	1,7/min	187,6
/images/co...	2	562	766	766	359	766	0,00%	1,2/min	46,6
/images/ha...	2	16437	31640	31640	1234	31640	0,00%	1,8/min	43,3
/images/bot...	2	1445	2469	2469	421	2469	0,00%	3,9/min	14,0
/images/ac...	2	921	1484	1484	359	1484	0,00%	4,3/min	55,4
/images/sy...	2	734	1109	1109	360	1109	0,00%	5,6/min	152,5
/images/ba...	2	1804	3219	3219	390	3219	0,00%	7,7/min	380,9
/sfGuardAut...	1	828	828	828	828	828	0,00%	36,2/min	5719,2
TOTAL	292	5814	453	26860	0	48656	0,00%	2,0/sec	18861,3

ANEXO 6. Lista de Chequeo.

Atributos de Calidad

Usabilidad

Nivel	Evaluación	Eval.	NP	Comentario
!	¿Se consideran otros idiomas para las instrucciones en la pantalla de forma adecuada?			
!	¿Hay ausencia de términos en idiomas diferentes mezclados?			
!	¿Es simple el vocabulario utilizado?			
!	¿Se proporciona tiempo suficiente para realizar las entradas por teclado?			
!	¿Se posibilita la opción de incrementar el tamaño de los caracteres?			
!	¿Hay algún tipo de asistencia para los usuarios que hacen uso del sistema por primera vez?			
!	¿Resulta fácil instalar el software?			
!	¿Proporciona el sistema un soporte apropiado a los usuarios más novatos?			
!	¿Se entienden la interfaz y su contenido?			
!	¿Resulta fácil especificar un objeto o una acción?			
!	¿Resulta fácil entender el resultado de una acción?			
!	¿Asiste el sistema al usuario de forma efectiva proporcionando el modo más efectivo de hacer las tareas en caso de que no haya una única forma de hacerlas?			
!!	¿Está diseñada la interfaz para facilitar la realización eficiente de las tareas de la mejor forma posible?			
!	¿Es fácil de utilizar el sistema en la realización de tareas?			
!	¿Facilita la interfaz de usuario un uso eficiente (para interfaces basadas en pantallas)?			
!	¿Actúa el sistema en la prevención de errores?			
!!	¿Actúa el sistema en la información de los errores?			
!	¿Se usan adecuadamente los modos de trabajo de los usuarios en el software?			
!	¿Se permite la utilización del ratón o el teclado?			
!	¿Permite al usuario interrumpir su tarea y continuar más tarde?			
!	¿Se utiliza mensajes y textos descriptivos?			

!!	¿Permite deshacer las acciones, e informar el estado?			
!!	¿Permite una cómoda navegación dentro del producto y una fácil salida de éste?			
!	¿Permite distintos niveles de uso del producto para usuarios con distintos niveles de experiencia?			
!	¿Se reconoce todas las tareas del software en cualquier momento?			
!!	¿Se proporciona información visual de dónde está el usuario, qué está haciendo y qué puede hacer a continuación?			
!	¿Proporciona funciones deshacer, rehacer?			
!	¿Existe atajos de teclado bien hechos?			
!	¿Se presenta al usuario la información que sólo necesita?			
!	¿Existen diferentes niveles de ayuda?			

Seguridad

Nivel	Evaluación	Eval.	NP	Comentario
!!	¿Las reglas de protección de Archivos de datos, las autoridades y códigos de identificación de usuario fueron establecidas por el dueño del sistema o asignado por una autoridad más alta?			
!!	¿El Acceso al control de protección está incorporado en el sistema?			
!	¿Todas las contraseñas del equipo de prueba y desarrolladores fueron borradas?			
!	¿Toda la privacidad, la libertad de información, sensibilidad, y consideraciones de la clasificación fueron identificadas, resueltas, y establecidas?			

Confiabilidad

Nivel	Evaluación	Eval.	NP	Comentario
!	¿Se recupera el software ante fallas?			
!	¿La recuperación ante fallas se realiza en el tiempo requerido para la aplicación?			

Portabilidad

Nivel	Evaluación	Eval.	NP	Comentario
!	¿Existe independencia del ambiente de hardware? (características particulares del hardware)			
!	¿Existe independencia del ambiente de software? (sistema operativo, lenguaje de programación)			

Notas:

LEYENDA:
Nivel: Importancia del aspecto a evaluar
E: Evaluación
NP: No Procede
Comentario: Es obligatorio en las respuestas negativas

Las evaluaciones serán:
-- Malo - propiedad no disponible. (0)
- Satisfactorio - propiedad parcialmente disponible. (2)
+ Bien - propiedad disponible. (4)
++ Excelente - propiedad muy bien implementada. (5)

El peso de la pregunta será:
!! Muy importante (5)
! Menos importante (3)

Glosario de Términos

A

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar las actividades; representa un área de responsabilidad y es candidata a ser tenida en cuenta para el control de la configuración. Un artefacto puede ser un modelo, un elemento de un modelo, o un documento. Véase trabajador, actividad.

C

Calidad: Calidad de software. Satisfacción de las necesidades de los usuarios.

Caso de uso**:** Es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. Secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas.

Código: Se refiere a las instrucciones contenidas en un programa, y entendibles por el ordenador. Las aplicaciones normales pueden tener miles de líneas de código que es necesario mantener y actualizar.

E

Estándar: Es cualquier tecnología aprobada por un comité formalizado.

F

Fallo: Manifestación de una falta.

Falta: Algo que está mal en un producto (modelo, código, documento, etc.).

Fase: Son los pasos en que se descomponen las metodologías. Cada fase puede o no estar subordinada a otra fase, pudiendo existir entre ellas relaciones de dependencia.

Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

H

Herramienta: Subprograma o módulo encargado de funciones específicas y afines entre sí para realizar una tarea. Una aplicación o programa puede contar con múltiples herramientas a su disposición. Por ejemplo, el corrector ortográfico puede ser una herramienta en una aplicación para redactar documentos, pero no es una aplicación en sí misma.

I

Interfaz: Una colección de operaciones que se usan para especificar el servicio de una clase o de un componente. Circuito electrónico que gobierna la conexión entre el hombre y el hardware, y los ayuda a intercambiar información de manera confiable.

M

Modelo: Representación de la realidad por abstracción.

P

Portabilidad: Característica de ciertos programas que les permite ser utilizados en distintos ordenadores sin que precisen modificaciones de importancia.

Procedimiento**:** Dentro de una aplicación, se denomina procedimiento al conjunto de instrucciones, controles, etc. que hacen posible la resolución de una cuestión específica. La impresión es un procedimiento, como lo es la incorporación de una imagen a un texto predeterminado, etc.

Proceso: Un proceso es una instancia de un programa. Actualmente los sistemas multitarea soportan la ejecución de múltiples procesos, dando la apariencia de que pueden correr simultáneamente (de forma concurrente). De hecho, sólo un proceso puede estar siendo ejecutado al mismo tiempo por el CPU (excepto los CPU con múltiples procesadores). Los procesos son creados, destruidos y comunicados entre sí por el sistema operativo. En Windows se pueden ver los procesos en ejecución desde el Administrador de Tareas.

R

Requerimiento

Funcionalidad requerida por un usuario para resolver un problema o satisfacer uno o varios objetivos.

S

Script: Pequeños programas incrustados en las páginas que nos permiten definir interactividades de cualquier tipo.

Seguridad: Seguridad es la cualidad de seguro, es decir, de estar libre y exentos de todo daño, peligro o riesgo.

Sistema Informático: Tiene por finalidad exclusiva y excluyente el almacenamiento, el procesamiento, la recuperación y la difusión de la información contenida en documentos de cualquier especie.

Conjunto u ordenación de elementos organizados para llevar a cabo algún Método, procedimiento o control mediante el proceso de información.

***Software*:** Software es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

T

Técnica: Conjunto de saberes prácticos o procedimientos para obtener un resultado. Requiere de destreza manual e intelectual, y generalmente con el uso de herramientas. Las técnicas se transmiten de generación en generación. La técnica nace en la imaginación y luego se llevan a la concreción, siempre de forma empírica. En cambio la tecnología surge de forma científica, reflexiva y con ayuda de las técnicas.

Tecnología: Abarca un conjunto de técnicas, conocimientos y procesos para el diseño y construcción de objetos para satisfacer necesidades humanas. La tecnología puede referirse a objetos que usa la humanidad (como máquinas, utensilios, hardware), pero también abarca sistemas, métodos de organización y técnicas, se basa en aportes científicos.

U

Usabilidad: La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y de ser atractivo para el usuario, en condiciones específicas de uso. La usabilidad también hacer referencia al grado de facilidad con que una aplicación, sitio Web, periférico o sistema, se adapta a sus usuarios. La usabilidad puede estar relacionada incluso a la Ergonomía, y a la potabilidad de un dispositivo.

Usuario: Persona que utiliza el software.