Universidad de las Ciencias Informáticas

Facultad 6

Título:

"Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica"

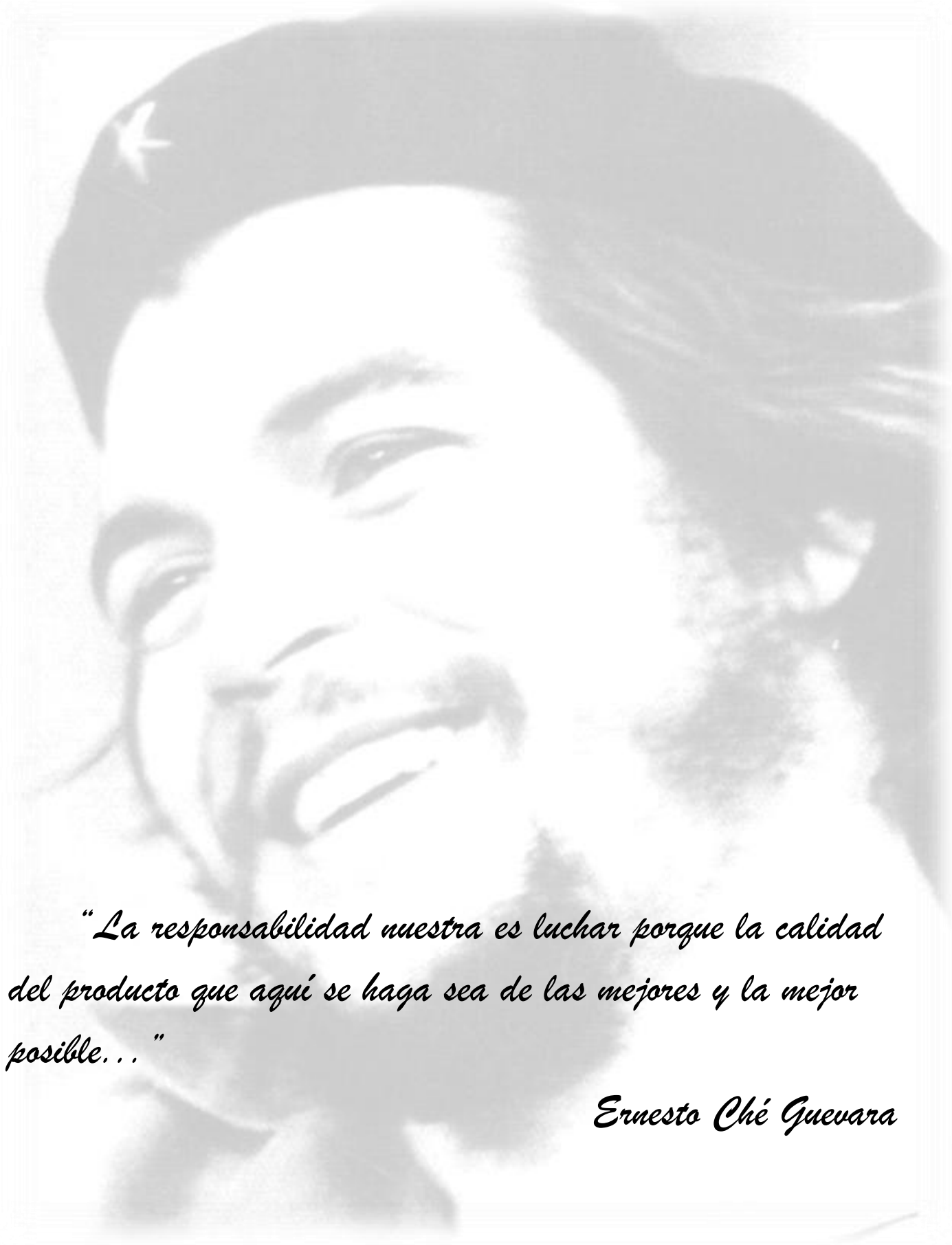
Trabajo de Diploma para optar por el título de Ingeniero Informático

Autores: Linet Lores Sánchez
Diana Monné Roque

Tutores: Ing. Martha Nieves Borrero
Ing. Lissete González Gallo

Ciudad de la Habana, Junio 2009

"Año del 50 Aniversario del Triunfo de la Revolución"



"La responsabilidad nuestra es luchar porque la calidad del producto que aquí se haga sea de las mejores y la mejor posible..."

Ernesto Ché Guevara

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 firmo la presente a los ____ días del mes de _____ del año _____.

Diana Monné Roque

_____

Firma del Autor

Linet Lores Sánchez

_____

Firma del Autor

Ing. Martha Nieves Borrero

_____

Firma del Autor

Ing. Lissete González Gallo

_____

Firma del Tutor

DATOS DE CONTACTO

DIPLOMANTES

Nombre: Linet Lores Sánchez
E-mail: lloress@estudiantes.uci.cu
linet.lores@gmail.com

Nombre: Diana Monné Roque
E-mail: dmonne@estudiantes.uci.cu
diana.monne@gmail.com

TUTORES

Nombre: Martha Nieves Borrero
E-mail: mnieves@uci.cu

Nombre: Lissete González Gallo
E-mail: lgallo@uci.cu

AGRADECIMIENTOS

A nuestras tutoras, Martha y Lissete por el apoyo que siempre nos brindaron.

A Roig, Delvis, Heydi, Lázaro, Adisley, el grupo de desarrollo del SIGM y a todos los que de una forma u otra nos ayudaron en el desarrollo de nuestra tesis.

A todos los profes que durante estos cinco años incentivaron nuestra formación profesional.

A nuestros amigos, por permitirnos disfrutar al máximo estos maravillosos años a su lado, por ser partícipes de nuestros logros, desconciertos, malos y buenos momentos, marcando una etapa inolvidable en nuestras vidas.

En general, agradecemos a todos aquellos que pusieron su granito de arena para que un día como hoy, pudiéramos dedicarles estas palabras.

DEDICATORIA

Diana

A mi mamita por su amor, confianza, dedicación, apoyo, por ser guía ejemplar y constante, por ser una madre excepcional y lo que más quiero de esta vida.

A mi otra mamá, mi tía Nena por ser tan maravillosa y por tener ese corazón tan grande, lleno de nobleza, ternura y amor para los suyos y ser para mí una madre más.

A mi tío Baquero, que mas que tío es como mi padre, por ser intransigente con mis estudios, por confiar en mí.

A mis hermanas mayores Dalgis y Danay, por su apoyo, consejos y todo el cariño que me brindan.

A Guillermo, por su apoyo, dedicación y confianza, por ser para él una hija y él para mí un maravilloso padre.

A Omar, por cada atención en estos 5 años.

A mi abuela Juana y mi prima Yami por la fe depositada en mí.

Al cuarteto Linet, Yislén, Alberto y Tomy, por saber llenar mi corazón con su cariño.

En fin a todas esas personitas que me hicieron esforzar para vencer los obstáculos.

Linet

A mi mamá, por brindarme siempre su apoyo incondicional, su cariño, su aliento, su fe; y darme las fuerzas que encontré para vencer cualquier obstáculo durante estos cinco años.

A mi abuelita, por ser aquella, mi fuente de inspiración, mi amuleto, el aliciente que nunca me faltó cuando en momentos de desesperanza necesité un incentivo para seguir adelante.

A mi tía Elena, por estar ahí, dispuesta a ayudarme en todo, por ser para mí como una madre en estos años de sacrificio y estudio intenso lejos de mi hogar.

A mi tío Carlos, por confiar en mí y ser una razón más para cada esfuerzo.

A Diana, Yuliet, Yislen, Alberto, Jany, Keila, Lisandra, Dunia y Leo, por ser aquellas personas de una forma u otra me brindaron su cariño, su ternura, su amor, iluminando mis días con su alegría.

RESUMEN

En el presente trabajo se dejó detalladamente documentado todo lo concerniente al proceso de pruebas de liberación realizado al Sistema Informático de Genética Médica (SIGM). Entre los tópicos que se trataron se encuentra una breve argumentación de los niveles, técnicas y métodos de prueba existentes, así como algunas herramientas para automatizar las pruebas con vista a determinar la calidad del software, distinguiendo de todos los aspectos mencionados, los más favorables al proceso de pruebas a desarrollar, y justificando su selección. Se pormenorizó la estrategia de trabajo y el procedimiento llevado a cabo en el proceso de las pruebas de liberación. Además se propone, fundamenta y describe la aplicación de las pruebas de carga y estrés. Finalmente se hizo un análisis evaluativo del proceso realizado para liberar el SIGM, cuyo propósito fundamental fue evaluar la calidad del mismo una vez realizadas las pruebas, las cuales arrojaron resultados puntuales para dar una valoración al respecto.

PALABRAS CLAVE

Calidad de software, pruebas de software, niveles de prueba, técnicas de prueba, métodos de prueba, herramientas de automatización de pruebas, proceso de pruebas de liberación, carga, estrés.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

El vertiginoso desarrollo actual de las Tecnologías de la Información y las Comunicaciones, ha acrecentado la necesidad de las compañías de destinar una mayor cantidad de sus presupuestos a las nuevas tecnologías aplicadas al desarrollo de su actividad. Todas estas innovaciones tecnológicas ameritan una exigente evaluación de la calidad de los productos.

Uno de los problemas que actualmente enfrenta la industria del software en el mundo lo constituye el tema de la calidad. ¿Cómo lograr la calidad de un software? ¿Cómo evaluar la calidad de un software?, son interrogantes que incentivan la preocupación de muchos especialistas, ingenieros, investigadores y comercializadores de software para obtener un producto fiable, eficaz, seguro, y funcionalmente operativo que cumpla con los requerimientos del cliente. Forma parte de sus objetivos estratégicos encontrar solución a esta problemática. Hoy día la calidad del software va más allá de un tratamiento centrado únicamente en la inspección y detección de errores, sino que ha escalado a una aproximación más sistemática y estricta, dada la importancia que ha adquirido la calidad en la producción de software.

Cuba, no queda exenta del desarrollo acelerado de la ciencia y las tecnologías de la información. Ello implica que las empresas informáticas enfrenten cada día, un reto para brindar una respuesta rápida, eficaz y con calidad a los clientes, que cada vez se vuelven más exigentes. Una muestra de estas entidades productoras de software la constituyen instituciones como: DeSoft, TranSoft, Softel y la Universidad de las Ciencias Informáticas, que, a pesar de ser un centro destinado a formar profesionales altamente calificados en esta rama, produce software y servicios informáticos.

El proceso de producción en la UCI se desarrolla por facultades, en las que se definen polos productivos. En la Facultad 6, existen dos polos: el Polo de Gestión de la Información Biomédica y el Polo de Bioinformática. Cada polo establece vínculos productivos con centros clientes para los cuales produce software. El Centro Nacional de Genética Médica (en lo sucesivo, CNGM) constituye uno de estos centros, el cual rige el progreso en el campo de la genética a través de una red nacional desarrollando diversos estudios en consonancia con esta tendencia mundial.

A raíz de los estudios genéticos realizados en el país, se detecta que no existe un sistema o herramienta que integre la gestión de los datos de una consulta de genética médica y de los estudios realizados por esta red. Dichos estudios, conjuntamente con las historias clínicas genéticas llevan

asociados un gran volumen de información, la cual se encuentran en formato duro, unido a que en términos de estadísticas, hay que recurrir a métodos convencionales que restan eficiencia, consistencia y rapidez del intercambio de la información generada. Con el objetivo de dar solución a los problemas expresados, a petición del CNGM, la Facultad 6 desarrolló el Sistema Informático de Genética Médica (en lo adelante, SIGM), diseñado para gestionar y automatizar toda la información asociada a una consulta de genética médica y a las investigaciones pertinentes de esta rama.

En el ámbito social, el SIGM tiene un impacto notable, debido a que permite que se puedan automatizar y almacenar de forma más eficiente las investigaciones sobre la genética humana, posibilitando llegar a un análisis acertado de la incidencia de las enfermedades genéticas en la población cubana. En adición, el sistema de Teleconsulta Genética provee al sistema de salud de Cuba, una herramienta que posibilite la discusión de casos a distancia, permitiendo acercar los servicios especializados de salud a la población cubana y no la población a los servicios, con el consiguiente ahorro de recursos para el paciente y el estado en sentido general.

El SIGM eleva la calidad de los servicios que hoy presta la red nacional de genética médica al poner la informática en función de la genética médica. Se brinda un servicio más eficiente al paciente, permitiendo darle un seguimiento constante traducido en el aumento de la calidad de vida de la población cubana.
Actualmente el SIGM se encuentra en su fase final de desarrollo; para su puesta en práctica se hace necesario que se compruebe el grado de concordancia entre las expectativas del cliente y el producto final.

Teniendo en cuenta la situación problémica existente se plantea como ***problema científico:***
¿Cómo verificar la calidad del Sistema Informático de Genética Médica?
Donde el ***objeto de estudio*** es: El proceso de pruebas de calidad de software.
Se estima como ***campo de acción***: Pruebas de liberación en el laboratorio de Calidad de la Facultad 6.
Como ***objetivo general*** se tiene: Desarrollar el proceso de pruebas de liberación al Sistema Informático de Genética Médica.
Desglosado en los sucesivos ***objetivos específicos***:

- Fundamentar las pruebas de software utilizadas en el proceso de liberación.

- Diseñar las pruebas de software para cada uno de los módulos a examinar y el sistema integrado.
- Aplicar las pruebas de software.
- Analizar los resultados del proceso de pruebas de liberación.

Para dar cumplimiento a los objetivos trazados, se plantean las siguientes ***tareas***:

- Estudio de las pruebas de software y herramientas para automatizar las pruebas.
- Realización del plan de pruebas.
- Revisión de la documentación del Software.
- Aplicación de listas de chequeo.
- Diseño de casos de prueba para Caja Negra.
- Realización de las pruebas a cada módulo.
- Aplicación de pruebas de Carga y Estrés.
- Análisis y evaluación de los resultados obtenidos.

El presente trabajo está estructurado en tres capítulos donde se exponen el desarrollo y los resultados de las pruebas realizadas:

- **Capítulo 1:** Fundamentación Teórica

  En el presente capítulo se aborda brevemente la metodología de desarrollo que sustenta el proceso de pruebas a realizar, los aspectos relacionados con dichas pruebas y la calidad del software. Para llevar a cabo el proceso de pruebas de liberación se realiza un estudio sobre los niveles, técnicas y métodos de prueba existentes, con vista a determinar la calidad del software, así como las herramientas para automatizar las pruebas. De todo ello se distinguen los más favorables al proceso de pruebas a desarrollar, justificando su selección.

- **Capítulo 2:** Diseño y aplicación de pruebas de liberación de software.

  Este capítulo describe las características del SIGM en cuanto a módulos, casos de uso y criticidad de los mismos. Detalla el proceso de diseño y realización de las pruebas de liberación al Sistema Informático de Genética Médica partiendo de la estrategia elaborada

para elaborar el mismo. Adicionalmente se describe detalladamente el proceso de automatización de las pruebas de carga y estrés con la herramienta JMeter.

- **Capítulo 3:** Análisis de los resultados.
  Todo proceso de pruebas requiere de un análisis final de los resultados obtenidos, de forma tal que se proporcione una evaluación del producto que se prueba teniendo en cuenta el seguimiento que se ha ido registrando de los defectos y fallos del sistema durante todo el proceso. Por cuanto en este capítulo se procura analizar las no conformidades encontradas según el elemento probado y técnica empleada. Se hace una valoración del producto de acuerdo a parámetros de calidad como la seguridad, portabilidad, usabilidad y confiabilidad, basándose en los resultados obtenidos.

CAPÍTULO 1

Fundamentación Teórica

Introducción

En el presente capítulo se aborda brevemente la metodología de desarrollo que sustenta el proceso de pruebas a realizar, los aspectos relacionados con dichas pruebas y la calidad del software. Para llevar a cabo el proceso de pruebas de liberación se realiza un estudio sobre los niveles, técnicas y métodos de prueba existentes, con vista a determinar la calidad del software, así como las herramientas para automatizar las pruebas. De todo ello se distinguen los más favorables al proceso de pruebas a desarrollar, justificando su selección.

1.1 Metodologías de desarrollo de software

Existen varias metodologías (o procesos) de desarrollo de software, cada una con sus propias características, aunque objetivamente persigan lo mismo. En dependencia de las particularidades de cada producto software se escoge la metodología a ser aplicada. Ejemplos de éstas son: RUP, XP, Open/UP.

Para el desarrollo del SIGM se empleó RUP (Rational Unified Process) como proceso de desarrollo, precisamente por sus propiedades. Así mismo, el proceso de pruebas de liberación realizado a este software se basa en esta metodología. A continuación algunos rasgos que evidencian las razones por las cuales se seleccionó la misma como proceso de desarrollo.

RUP se divide en 4 fases el desarrollo del software:

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

Cada una de estas etapas es desarrollada mediante un ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. [16]

El ciclo de vida que se desarrolla por cada iteración, a través de los flujos de trabajo es:

- Modelamiento del Negocio: Se comprenden las necesidades y los procesos del negocio.

- Levantamiento de requerimientos: Se trasladan las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Se refinan los requerimientos y trasladándolos dentro de la arquitectura de software.
- Implementación: Se crea un software que se ajuste a la arquitectura y funcione.
- Pruebas: Se verifica que el comportamiento requerido es el correcto y que todo lo solicitado está presente.
- Despliegue: Hacer todo lo necesario para la salida del proyecto.
- Configuración y administración del cambio: Se guardan todas las versiones del proyecto.
- Administrando el proyecto: Se administran horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.

RUP define para el flujo de trabajo de Pruebas 4 roles fundamentales [4]:

- Jefe de Pruebas

  Es el responsable del éxito de la prueba, Este rol es el defensor de prueba y de la calidad, planifica y administra los recursos, resuelve los problemas que impidan las pruebas.
- Analista de pruebas

  Es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de las mismas y el resultado en cada ciclo de pruebas, evaluando la calidad total experimentada como un resultado de las actividades de prueba. Este rol lleva la responsabilidad para representar apropiadamente las necesidades de los stakeholders que no tienen representación regular y directa en el proyecto.
- Diseñador de pruebas

  Es el responsable de definir el método de prueba y asegurar su implementación exitosa. El rol incluye identificar técnicas apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y encauzar los recursos correspondientes para las pruebas.
- Probador

  Conduce las pruebas necesarias y el registro del resultado de las mismas.

En el caso del proceso de pruebas de liberación en cuestión, se desempeñaron todos los roles mencionados.

RUP posee tres características principales:

- Dirigido por casos de uso, estos nos proporcionan un hilo conductor ya que avanza a través de una serie de flujos de trabajos que parten de ellos.
- Centrado en la arquitectura, muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Iterativo e incremental, es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. [17]

1.2 Calidad del Software

En un mundo globalizado, donde las organizaciones se ven enfrentadas a competencias de nivel mundial, la calidad se convierte en un importante punto diferenciador, además de aumentar la satisfacción general del cliente, disminuir costos y optimizar los recursos. Los productos o servicios que ostentan certificados de calidad son preferidos por los compradores porque trasmiten seguridad y confianza. Esto también constituye un atributo de valor para las estrategias de comercialización en el exterior.

La calidad en las empresas ha evolucionado, si se analiza desde los inicios de los procesos de industrialización a mediados del siglo XIX hasta cerca de 1940, la calidad se relacionaba con la inspección en los productos con el propósito de detectar errores, de esta fecha hasta los años 80 el control de calidad se convirtió en un ejercicio de control estadístico cuyo propósito era impedir que el producto defectuoso llegara al cliente y a partir de los 80 se inician procesos de gestión de calidad total, buscando garantizar la calidad por medio de la planificación y la creación de modelos de calidad de forma permanente. El interés por la calidad del software asciende continuamente, debido a las exigencias cada vez más crecientes de los clientes y la competitividad entre las empresas.

Existen varias definiciones de la *Calidad del Software* expresadas por diversas compañías y personalidades destacadas en este sector, tales como:

“Conjunto de características de un producto o servicio que le confieren su aptitud para satisfacer necesidades expresadas e implícitas”. [1]

"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". [2]

"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". [3]

1.3 Pruebas de Software

Para contribuir con la calidad del software es recomendable que el producto software vaya siendo evaluado a medida que se va construyendo. Posteriormente, se hace necesario llevar cabo, paralelo al proceso de desarrollo, un proceso de evaluación y comprobación de los distintos productos o modelos que se van generando, en el que participarán desarrolladores y clientes.
Las pruebas de software constituyen un pilar indispensable para evaluar y determinar la calidad de un software. Concretamente se puede definir pruebas de software como:

- El proceso de ejecución de un programa con la intención de descubrir errores previos a la entrega al usuario final.
- Una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones específicas, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.[4]

Tienen como objetivos:

- Encontrar y documentar los defectos que pueden afectar la calidad del software.
- Verificar que el software trabaje como fue diseñado.
- Validar y probar los requisitos que debe cumplir el software.
- Validar que los requisitos fueron implementados correctamente.

Otros conceptos importantes relacionados con las pruebas de software son:

Validación

Es la operación de asegurar que los requerimientos funcionales y no funcionales están correcta y completamente implementados en el producto final. [14]

Verificación

Es la operación de establecer las especificaciones y entradas adecuadas a una actividad y la de

establecer que las salidas de las actividades son correctas y consistentes con las especificaciones y la entrada. [14]

Un buen proceso de pruebas sustentado en la validación y verificación, permitirá mejorar y asegurar la calidad del desarrollo durante toda esta etapa. Permite, un mayor control y una identificación temprana de errores y problemas, así como una reducción notable de los costes para subsanar estos errores.

Defecto

Un defecto en el software es un proceso, una definición de datos o un paso de procesamiento incorrectos en un programa. [15]

Fallo

La incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados. [15]

Error

La diferencia entre un valor calculado, observado o medio y el valor verdadero, especificado o teóricamente correcto.

- Un defecto
- Un resultado incorrecto
- Una acción humana que conduce a un resultado no válido [15]

1.4 Niveles de Prueba

A la hora de evaluar dinámicamente un sistema software se debe comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Las pruebas se aplican durante todo el ciclo de desarrollo del software para diferentes objetivos y en distintos niveles de trabajo, dentro de estos se distinguen:

- **Pruebas de Desarrollador**

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para las pruebas 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. [4]

- **Pruebas Independientes**

Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores. El objetivo de estas pruebas 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[1], que son pruebas basadas en las necesidades y preocupaciones de los stakeholders. [4]

- **Pruebas de Unidad**

Se enfocan en un programa o un componente que desempeña una función específica que puede ser probada y que se asegura que funcione tal y como lo define la especificación del programa. Los programadores siempre prueban el código durante el desarrollo, por lo que las pruebas unitarias son realizadas solamente después de que el programador considera que el componente está libre de errores.

Al tratar software OO[2] cambia el concepto de unidad. El encapsulamiento dirige la definición de clases y objetos. Esto significa que cada clase e instancia de clase (objeto) empaqueta los atributos (datos) y las operaciones (también conocidas como métodos o servicios) que manipulan estos datos. Por lo tanto, en vez de módulos individuales, la menor unidad a probar es la clase u objeto encapsulado. Una clase puede contener un cierto número de operaciones, y una operación particular puede existir como parte de un número de clases diferentes.

El significado de prueba de unidad cambia ampliamente frente al concepto general visto antes. De esta manera, la *prueba de clases* para el software OO es el equivalente a la prueba de unidad para software convencional. A diferencia de la prueba de unidad del software convencional, la cual tiende a centrarse en el detalle algorítmico de un módulo y los datos que fluyen a lo largo de la interfaz de éste, la prueba de clases para software OO está dirigida por las operaciones encapsuladas en la clase y el estado del comportamiento de la clase. [5]

- **Pruebas de Integración**

[1] Personas u organizaciones que están activamente implicadas en el negocio ya sea porque participan en él o porque sus intereses se ven afectados con los resultados del proyecto.

[2] Orientado a Objeto

Su objetivo es identificar errores introducidos por la combinación de programas o componentes probados unitariamente, además, verificar que las especificaciones de diseño sean alcanzadas. Componentes individuales son combinados con otros componentes para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente. No son verdaderamente pruebas de sistema porque los componentes no están implementados en el ambiente operativo. [5]

En este nivel se asegura que las interfaces y ligas entre las partes del sistema trabajen apropiadamente. Antes de las pruebas de integración, los componentes tuvieron que haber pasado sus pruebas individuales, por lo que, el enfoque ahora es sobre el flujo de control entre los módulos, y sobre los datos que son intercambiados entre ellos. A menudo hay una tendencia a intentar una integración *no incremental*; es decir, a combinar todos los módulos y probar todo el programa en su conjunto. El resultado puede ser un poco caótico con un gran conjunto de fallos y la consiguiente dificultad para identificar el módulo (o módulos) que los provocó.

En contraposición, se puede aplicar la integración *incremental* en la que el programa se prueba en pequeñas porciones en las que los fallos son más fáciles de detectar. Existen dos tipos de integración incremental, la denominada *ascendente* y *descendente*.

- Ascendente: Se empieza con los módulos de nivel inferior, y se verifica que los módulos de nivel inferior llaman a los de nivel superior de manera correcta, con los parámetros correctos.
- Descendente: Se empieza con los módulos de nivel superior, y se verifica que los módulos de nivel superior llaman a los de nivel inferior de manera correcta, con los parámetros correctos.

En el caso del software OO no se tiene una estructura de control jerárquica, por lo que las estrategias convencionales de integración ascendente y descendente poseen un significado poco relevante en este contexto. Generalmente se pueden encontrar dos estrategias diferentes de pruebas de integración en sistemas OO.

- Prueba basada en hilos: Integra el conjunto de clases necesario para responder a una entrada o evento del sistema. Cada hilo se integra y prueba individualmente.
- Prueba basada en el uso: Comienza la construcción del sistema integrando y probando aquellas clases (llamadas clases independientes) que usan muy pocas de las clases. Después de probar las clases independientes, se comprueba la próxima capa de clases, llamadas clases dependientes, que usan las clases independientes. Esta secuencia de capas de pruebas de clases dependientes continúa hasta construir el sistema por completo.

La prueba basada en hilos proporciona una estrategia más ordenada para realizar la prueba que la prueba basada en el uso. La primera, suele aplicarse utilizando los diagramas de secuencia de objetos que diseñan cada evento de entrada al sistema. [5]

- **Pruebas al Sistema**

Son usualmente conducidas para asegurar que todos los módulos trabajan como sistema sin error. Es similar a la prueba de integración pero con un alcance mucho más amplio. Las pruebas del sistema examinan qué tan bien el sistema cumple con los requerimientos de la organización y su utilidad, seguridad y desempeño. También se realizan estas pruebas a la documentación del sistema.
Concretamente se debe comprobar que:

- Se cumplen los requisitos funcionales establecidos.
- Sea correcto el funcionamiento y rendimiento de las interfaces hardware, software y de usuario.
- Sea apropiada la documentación de usuario.
- Se verifique el rendimiento y respuesta en condiciones límite y de sobrecarga.

Como en el caso del software convencional, la validación del software OO se centra en las acciones visibles del usuario y las salidas del sistema reconocibles por éste. Para asistir en la determinación de casos de prueba de sistema, el ejecutor de la prueba debe basarse en los casos de uso que forman parte del modelo de análisis. El caso de uso brinda un escenario que posee una alta probabilidad con errores encubiertos en los requisitos de interacción del cliente. Los métodos convencionales de prueba de caja negra, pueden usarse para dirigir estas pruebas. [5]

- **Pruebas de Aceptación**

Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

Las pruebas de aceptación son realizadas en dos etapas, una llamada *alpha*, en la cual los usuarios prueban el sistema usando datos de prueba, y la segunda llamada *beta* en la cual los usuarios empiezan a utilizar el sistema con los datos reales, pero siendo aun monitoreados cuidadosamente previendo que se presenten errores.

Es muy recomendable que las pruebas de aceptación se realicen en el entorno en que se va a explotar el sistema (incluido el personal que lo maneje). En caso de un producto de interés general, se realizan pruebas con varios usuarios que reportarán sus valoraciones sobre el producto. [5]

1.4.1 Niveles seleccionados

En la etapa del ciclo de vida en que se encuentra el software, ya se ha transitado por los niveles precedentes, por tanto las pruebas independientes y de desarrolladores no forman parte del proceso de liberación concebido en la facultad 6, sino más bien como parte del ciclo de desarrollo del software. En el caso de las pruebas de aceptación, se realizan una vez ejecutadas las pruebas de liberación y emitida el acta donde conste que el software está listo para poner en marcha en la entidad que lo solicita; son responsabilidad del equipo de desarrolladores.

Los niveles que se han seleccionado para llevar a cabo el proceso de liberación del SIGM son:

- Pruebas de integración
- Pruebas al sistema

Las pruebas de integración se rigen por la estrategia de las pruebas basadas en hilos, pues proporcionan un resultado más organizado y consecuente con lo que se persigue verificar. Las pruebas al sistema a pesar de ser un nivel superior al de Integración, se ve con este último estrechamente relacionado. Dado que las pruebas al sistema básicamente son similares a las pruebas de Integración pero con un alcance mucho más amplio, a medida que se integran los módulos, se va probando el sistema como un todo, conjuntamente con la documentación que lo sustenta.

1.5 Técnicas de prueba

Cada nivel de prueba engloba una técnica de prueba específica según los atributos de calidad que se deseen verificar con las pruebas al software.

- **Pruebas de Funcionalidad**

Entre las técnicas de pruebas que se realizan en un sistema está la que evalúa la funcionalidad de éste. Se pueden dilucidar distintos tipos de prueba, según los parámetros de evaluación que abarca la técnica:

- **Prueba funcional**

Objetivo:

Asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Metas**:**

Verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio y la apropiada aceptación de datos.

Enfoque**:**

Los requisitos funcionales y las reglas del negocio.

Método**:**

Caja Negra: Se ejecuta cada caso de uso, flujo de caso de uso, o función, usando datos válidos e inválidos, para verificar lo siguiente:

- ✓ Que se aplique apropiadamente cada regla de negocio.
- ✓ Que los resultados esperados ocurran cuando se usen datos válidos.
- ✓ Que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos. [6]

- **Prueba de seguridad**

Objetivo:

- ✓ Nivel de seguridad de la aplicación: Verifica que un actor sólo pueda acceder a las funciones y datos que su usuario tiene permitido.
- ✓ Nivel de Seguridad del Sistema: Verifica que sólo los actores con acceso al sistema y a la aplicación están habilitados para accederla.

Áreas:

- ✓ Seguridad del sistema, incluyendo acceso a datos o funciones de negocios.
- ✓ Seguridad del sistema, incluyendo ingresos y accesos remotos al sistema.

Garantiza:

- ✓ Que los usuarios están restringidos a funciones específicas o su acceso está limitado únicamente a los datos que está autorizado a acceder.
- ✓ Que sólo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema.

Técnicas:

- ✓ Identificar cada tipo de usuario y las funciones y datos a los que se debe autorizar.
- ✓ Crear pruebas para cada tipo de usuario y verificar cada permiso, creando transacciones específicas para cada tipo de usuario.
- ✓ Modificar tipos de usuarios y volver a ejecutar las pruebas.

Criterio de completitud.

Para cada tipo de usuario conocido, las funciones y datos apropiados y todas las transacciones funcionan como se esperaba. [6]

- **Prueba de volumen**

Objetivo:

Verificar que la aplicación funciona adecuadamente bajo los siguientes escenarios de volumen:

1.-Máximo (actual o físicamente posible) número de clientes conectados (o simulados), todos ejecutando la misma función (peor caso de desempeño) por un período extendido.

2.-Máximo tamaño de base de datos (actual o escalado) y múltiples consultas ejecutadas simultáneamente.

Descripción:

Verificar que la aplicación funciona adecuadamente bajo los siguientes escenarios de volumen:

1.-Determinan la cantidad de datos con la cual el sistema falla.

2.-Carga máxima que el sistema soporta en un período dado.

Técnica:

- ✓ Usar múltiples clientes, corriendo las mismas pruebas o pruebas complementarias para producir el peor caso de volumen por un período extendido.
- ✓ Utilizar un tamaño máximo de base de datos (actual o con datos representativos) y múltiples clientes para correr consultas simultáneamente para períodos extendidos.

Criterio de completitud:

Todas las pruebas planeadas han sido ejecutadas y los límites especificados en el sistema se han conseguido o excedido sin que el sistema falle. [6]

- **Pruebas de Usabilidad**

El término usabilidad puede referirse a dos conceptos:

1. La usabilidad como atributo de calidad de un sistema. Por ejemplo: “Es necesario desarrollar una intranet usable”.
2. La usabilidad como un proceso que se aplica durante un proyecto. Esto también se conoce como “ingeniería de uso”, o “diseño centrado en el usuario”. [7]

Usabilidad como atributo

La usabilidad como atributo de calidad, la norma ISO 9241-11 la define como: “La medida en que un producto puede ser utilizado por usuarios específicos, para alcanzar metas específicas, de forma efectiva, eficiente y satisfactoria, dentro de un contexto de uso”. [7] Aunque lo anterior parezca algo redundante, tiene varios puntos a resaltar, y se deben tomar en cuenta para crear sistemas usables:

1. Los usuarios de un sistema se especifican.
2. Los usuarios tienen un conjunto de metas.
3. Un sistema debe permitir a los usuarios lograr dichas metas de forma eficiente, con lo cual quedarán satisfechos.
4. Un sistema se utiliza dentro de un contexto en particular (por ejemplo, bajo un ambiente específico).

Jakob Nielsen, considerado el padre de la usabilidad, la definió como el atributo de calidad que mide lo fáciles de usar que son las interfaces Web. Es decir un sitio Web usable es aquel en el que los usuarios pueden interactuar de la forma más fácil, cómoda, segura e inteligentemente posible.

¿Cómo saber si un sistema es usable?

La esencia de las pruebas de usabilidad radica en la observación objetiva del usuario, mientras utiliza el sistema para llevar a cabo acciones reales. En las pruebas de usabilidad, los observadores no le

enseñan a los participantes cómo utilizar el sistema, y tampoco contestan preguntas; debe ser como si los participantes estuvieran solos.

Hay tres factores claves para realizar las pruebas de usabilidad:

- Los participantes deben ser usuarios actuales o futuros del sistema. Un error común, es dejar que los gerentes prueben el sistema, en lugar de los operadores, que son los que realmente van a utilizar el sistema, una vez que esté en producción.
- Los participantes deben intentar realizar las tareas que normalmente realizarían con el sistema. Es crucial que estas tareas sean realistas.
- El contexto bajo el que se realiza la prueba, debe ser lo más cercano posible al contexto real en que el sistema se utilizará. [7]

La Usabilidad como Proceso

Para crear un sistema usable, se recomienda seguir un proceso de diseño centrado en el usuario, donde se involucra a los usuarios durante el desarrollo del software, y se realizan pruebas de usabilidad en diversas etapas del proyecto. El diseño centrado en el usuario es un conjunto de actividades y técnicas, cuyo objetivo es crear un sistema usable. Tales actividades se pueden incluir a través de las etapas de desarrollo de un software, desde el conceptual, hasta las pruebas finales, y aun en el mantenimiento y soporte. No es una receta paso a paso. Es más una filosofía, que una metodología. Se trata de, genuinamente buscar la creación de algo que pueda utilizarse por una audiencia real; y recordar que el sistema es para sus usuarios, y no para sus desarrolladores. [7]

- **Pruebas de Confiabilidad**

La confiabilidad del software se refiere a la precisión con la que una aplicación proporciona, sin errores, los servicios que se establecieron en las especificaciones originales. El diseño para favorecer la confiabilidad, además de referirse al tiempo de funcionamiento de la aplicación antes de que se produzca algún error, está relacionado también con la consecución de resultados correctos y con el control de la detección de errores y de la recuperación para evitar que se produzcan errores. [8] Según [MIL-STD 721C][3] es una medida del grado de operabilidad y capacidad de un sistema para prestar el

[3] Definitions of Terms For Reliability and Maintainability

servicio requerido en cualquier momento de su tiempo de misión, suponiendo su disponibilidad en el instante inicial.

Las técnicas de crecimiento de fiabilidad del software (cuyo acrónimo en inglés es SRGM - Software Reliability Growth Models) se utilizan para estimar la tasa de fallos y la fiabilidad de un sistema de software. La necesidad de este tipo de medidas se enmarca dentro del objetivo de cuantificar la confiabilidad de un sistema software, que hace referencia a la "calidad del servicio prestado" de forma que se pueda confiar de una forma justificada en su servicio. La confiabilidad no se mide directamente, sino a través de sus atributos, los cuales son: fiabilidad, disponibilidad, mantenibilidad y seguridad. [9]

Técnicas SRGM

Aplicando técnicas de SRGM durante la fase de pruebas del desarrollo de un sistema de software, se podrá predecir el número de fallos que tendrá un sistema en la fase de producción, de forma que el gestor del proyecto dispondrá de una útil herramienta de decisión para:

1. Establecer si el conjunto de pruebas al que ha sido sometido el sistema ha resultado suficiente, de forma que se pueda autorizar su paso a producción.
2. Reducir la posibilidad de que ocurra una incidencia grave en producción.
3. Estimar la tasa de fallos del sistema.
4. Estimar el momento, durante la fase de pruebas, en que se alcanzan los objetivos de fiabilidad del sistema. Por ejemplo, se puede haber determinado en los requisitos del sistema la calidad del software como: "Después de las pruebas, con un nivel de confianza del 95% deben quedar menos de 10 errores residuales en el sistema". Utilizando estas técnicas se podría estimar cuándo se alcanzará este objetivo. [9]

Existen diversos tipos de prueba para verificar la confiabilidad de un sistema.

- **Pruebas de estrés**

  Las pruebas de estrés consisten en la simulación de grandes cargas de trabajo para observar de qué forma se comporta la aplicación ante situaciones de uso intenso.

  - ✓ **Pruebas de estrés de componentes**

    Con las pruebas de estrés de los componentes, se aíslan los servicios y componentes que conforman el sistema, se infieren los métodos de navegación, de funcionamiento y

de interfaz de estos servicios y componentes, y se crea un cliente de prueba que llame a dichos métodos.

Para aquellos métodos que tienen acceso a un servidor de base de datos o a cualquier otro componente, puede crear un cliente que proporcione datos simulados en el formato previsto. El equipo de prueba inserta datos simulados una y otra vez mientras observa los resultados. La idea es forzar cada componente de forma aislada más de lo que la aplicación podría experimentar en condiciones normales. [8]

- ✓ **Pruebas de estrés de integración**

Después de forzar cada componente individual, deberá someter a una situación de estrés a toda la aplicación con todos sus componentes y servicios. Las pruebas de estrés de integración están íntimamente relacionadas con las interacciones con otras estructuras de datos, procesos y servicios tanto de los componentes internos como de otros servicios externos de la aplicación.

Las secuencias de comandos de prueba deberán probar la aplicación de acuerdo con el uso previsto. Por ejemplo, si la aplicación muestra una página Web que un 99% de los clientes simplemente visitará y en la que sólo un 1% comprará realmente, tiene sentido proporcionar secuencias de comandos de prueba que fuercen la búsqueda y las distintas funciones de exploración. Por supuesto, la cesta de la compra también debe comprobarse, pero el uso previsto sugiere que la mayoría de las pruebas deberían centrarse en las funciones de búsqueda. [8]

Se intenta prolongar siempre la duración de las pruebas, tanto como se lo permitan el calendario y el presupuesto. En lugar de realizar pruebas durante unos cuantos días o una semana, prolongue el período de pruebas a un mes, un trimestre o un año y observe cómo funciona la aplicación durante un período de tiempo más largo.

- **Pruebas de destrucción aleatorias**

Una de las formas más sencillas de probar la confiabilidad es utilizar datos de entrada aleatorios. Este tipo de pruebas intenta por todos los medios bloquear la aplicación o que ésta produzca errores; para ello, se proporcionan datos ilógicos y falsos. Los datos de entrada pueden ser eventos del mouse (ratón) o del teclado, secuencias de mensajes del programa, páginas Web, cachés de datos o cualquier otra condición de entrada que pueda

introducirse en la aplicación. Deberá utilizar pruebas de destrucción aleatorias para comprobar las rutas de errores importantes y poner de manifiesto errores de programación del software. Este tipo de pruebas mejora la calidad del código ya que da lugar a errores que permiten examinar el control de los errores devueltos.

Las pruebas aleatorias pasan por alto de forma intencionada cualquier especificación del comportamiento del programa. Si se interrumpe la aplicación, no se ha superado la prueba. Si no se interrumpe la aplicación, la prueba se ha superado. [8]

- **Pruebas Estructurales**

  Utilizan el método de Caja Blanca próximo a tratar, son también conocidas como "pruebas basadas en código", donde se enfocan en probar cada una de las estructuras de código, para que su comportamiento sea el esperado.

  Son las pruebas donde se conoce la estructura interna del componente a probar, y se efectúa una prueba sobre dicha estructura. En el caso de una aplicación web también se revisa la estructura interna de los links y otros elementos. [8]

- **Pruebas de Rendimiento**

En la ingeniería del software, las pruebas de rendimiento o desempeño como también se conoce, son las que se realizan, desde una perspectiva, para determinar cuán rápido realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Las pruebas de rendimiento son un subconjunto de la ingeniería de pruebas, una práctica informática que se esfuerza por mejorar el rendimiento, incluyéndose en el diseño y la arquitectura de un sistema, antes incluso del esfuerzo inicial de la codificación.

Las pruebas de desempeño se basan en comprobar que el sistema puede soportar el volumen de carga definido en la especificación, es decir, la eficiencia. Se evalúan los resultados de una prueba para un caso de uso comparándolo con varias ejecuciones de la misma y se examinan las estadísticas resumidas recaudadas para un caso de uso en busca de indicadores de variabilidad de las respuestas del sistema. Para su diagnóstico, los ingenieros de software utilizan herramientas que monitorean el flujo de eventos de la aplicación y arrojan resultados estadísticos que tasan el desempeño del sistema,

controlando los tiempos de respuestas, la carga de usuarios, el uso de recursos, y comportamiento del software bajo determinadas condiciones de trabajo a las que se someta.

Es fundamental para alcanzar un buen nivel de rendimiento de un nuevo sistema, que los esfuerzos en estas pruebas comiencen en el inicio del proyecto de desarrollo y se amplíe durante su construcción. Cuanto más se tarde en detectar un defecto de rendimiento, mayor es el coste de la solución. Esto es cierto en el caso de las pruebas funcionales, pero mucho más en las pruebas de rendimiento, debido a que su ámbito de aplicación es de principio a fin. Para medir el rendimiento se realizan diferentes tipos de pruebas:

- **Pruebas de carga**

  Este es el tipo más sencillo de pruebas de rendimiento. Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación.

- **Prueba de estrés**

  Esta prueba se utiliza normalmente para hacer colapsar la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se colapsa. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores a determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

- **Prueba de estabilidad**

  Esta prueba normalmente se hace para determinar si la aplicación puede aguantar una carga esperada continuada. Generalmente se persigue determinar si hay alguna fuga de memoria en la aplicación.

- **Pruebas de picos**

La prueba de picos, como el nombre sugiere, trata de observar el comportamiento del sistema variando el número de usuarios, tanto cuando bajan, como cuando tiene cambios drásticos en su carga.

- **Pruebas de Soportabilidad**

Esta técnica de prueba se asegura de que el colocar piezas de código a correr en una computadora del cliente pueda ser hecho fácilmente.

Verifica la operación del elemento de prueba sobre diversas configuraciones del software y de hardware.

Hay dos tipos:

- ✓ De Configuración: Se enfocan en evaluar aquellos elementos configurados para diferentes hardware y/o configuraciones de software. Pueden implementarse como pruebas de rendimiento del sistema.
- ✓ De Instalación: Se enfoca en evaluar que el elemento a probar se instala como se indica, en diferentes configuraciones de hardware y software.

- **Pruebas de Configuración**

Se enfocan en evaluar las diferentes variaciones de una aplicación integrada, contra sus requerimientos de configuración.

Meta:

Hacer que la aplicación falle en cumplir sus requerimientos de configuración, de manera que los defectos escondidos sean identificados, analizados, arreglados, y prevenidos en el futuro.

Objetivos:

- ✓ Validar la aplicación parcialmente.
- ✓ Causar fallas para identificar defectos no identificados en las pruebas unitarias y en las pruebas de integración.
- ✓ Reportar estas fallas a los desarrolladores.
- ✓ Determinar el efecto de añadir o modificar los recursos de hardware.
- ✓ Determinar la configuración óptima del sistema.

Precondiciones:

- ✓ Los requerimientos de configuración pueden ser especificados.
- ✓ Existen múltiples variables.
- ✓ Los componentes pasaron las pruebas unitarias.

- ✓ El equipo está entrenado para aplicar estas pruebas.
- ✓ El ambiente de prueba está listo.

Poscondiciones:

- ✓ Al menos existe un test suite por cada requerimiento.
- ✓ Estas suites corren con éxito en la configuración apropiada.[10]

- **Instalación**

Primer propósito:

Es asegurarse de que el software se puede instalar bajo diversas condiciones tales como una nueva instalación, una actualización, e instalaciones personalizadas o completas bajo condiciones normales o anormales. Las condiciones anormales incluyen insuficiente espacio en disco, escasos privilegios de crear directorios, etcétera.

Segundo propósito:

Es verificar que, una vez que esté instalado, el software funciona correctamente. Esto generalmente significa correr un número de pruebas que fueron desarrolladas en las Pruebas de Funcionalidad. [11]

1.5.1 Selección de las técnicas a emplear

El conjunto de pruebas seleccionadas se compone fundamentalmente por:

- Pruebas de Funcionalidad
- Pruebas de Confiabilidad
- Pruebas de Rendimiento

Para las pruebas de funcionalidad, mediante las pruebas funcionales se trata de garantizar que el software trabaje funcionalmente como se especificó inicialmente por el cliente. Con las pruebas de seguridad se verifica que un usuario sólo pueda acceder a las funciones y datos que tiene permitido, o sea, para cada tipo de usuario conocido, las funciones y 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, comprobando el desempeño, resistencia y rendimiento del sistema. Cuando se prueba la funcionalidad del sistema se aplica de manera implícita la destrucción aleatoria, que es un tipo de prueba dentro de la técnica de confiabilidad.

Como se ha podido observar, las técnicas de prueba no son entes que deben verse desligados. Son más bien un conjunto integrado de acciones que combinadas, de forma general, miden, verifican, y evalúan, disímiles parámetros de calidad de software, para así lograr el objetivo trazado con la realización de las pruebas.

1.6 Métodos de Prueba

Existen métodos de prueba independientemente de la técnica que se utilice o el nivel en que se enmarquen estas técnicas. Estos métodos proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas, agrupándose en:

- Método de Caja Blanca o Estructural: Se basa en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- Método de Caja Negra o Funcional: Se realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.[5]

1.6.1 Método de Caja Blanca

El objetivo del método es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa. [5] Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código.

Este método garantiza que:

- ✓ Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- ✓ Se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez.

Puede ser impracticable realizar una prueba exhaustiva de todos los caminos de un programa. Por ello se han definido distintas técnicas de pruebas para implementar este método, que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba.

- **La prueba del camino básico:** Permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa. Constituye la técnica más usada dentro del método de Caja Blanca.
- **La prueba de condición:** Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
- **La prueba de flujo de datos**: Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- **La prueba de bucles:** Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

1.6.2 Método de Caja Negra

También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. Se centran principalmente en los requisitos funcionales del software.

Para seleccionar el conjunto de entradas y salidas sobre las cuales trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Entonces, dado que la prueba exhaustiva es imposible, el objetivo final es encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible. [5]

En esencia este método permite encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Al igual que ocurre con las técnicas de Caja Blanca, para confeccionar los casos de prueba de Caja Negra existen distintos criterios; algunos de ellos son:

- **Técnica de la Partición de Equivalencia**: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del Análisis de Valores Límites**: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de Grafos de Causa-Efecto**: Permite al encarga do de la prueba validar complejos conjuntos de acciones y condiciones.

De éstas, la técnica más conocida por su uso es la de Partición de Equivalencia, la cual se detalla en el siguiente epígrafe.

1.6.2.1 Técnica de la Partición de Equivalencia

La partición de equivalencia es una técnica del método de prueba de Caja Negra que divide el campo de entrada de un programa en *variables* con juegos de datos de entrada y salida. En esencia, esta técnica intenta dividir el dominio de entrada de un programa en un número finito de *variables de equivalencia.* De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada variable es equivalente a una prueba realizada con cualquier otro valor de dicha variable.

Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Éstas se identifican examinando cada condición de entrada (normalmente una frase en la especificación) y dividiéndola en dos o más grupos. Se definen dos tipos de variables de equivalencia, las *válidas*, que representan entradas válidas al programa, y las *no válidas*, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

Esta técnica de Caja Negra es precisamente la que se utiliza para probar la funcionalidad del SIGM y diseñar los casos de prueba. Es preciso dejar explícito la técnica y el método escogido para realizar las pruebas funcionales, de manera que se entienda la elección. 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, costos, y se obtenga un resultado satisfactorio.

En el caso del método de Caja Blanca, es más apropiado para ser utilizado por los programadores del software, ya que está directamente relacionado con el código, y se debe tener un profundo conocimiento del código que se analiza. No es apropiado utilizar este método para probar las funcionalidades del sistema, que es justamente lo que se pretende verificar antes de que el software sea liberado, que cumpla con los requerimientos funcionales establecidos por el cliente y tenga un desempeño y rendimiento adecuados. De ahí que sea justa la elección del método Caja Negra con su correspondiente técnica de la Partición de Equivalencia.

1.7 Proceso de Pruebas de Liberación

En la facultad 6 el proceso de pruebas de liberación de un software es aquel procedimiento que engloba un conjunto de pruebas para determinar la calidad del software cuando éste se encuentra ya en su fase terminal, es decir, cuando el producto está concluido y listo para poner en funcionamiento en la entidad que lo demanda. Durante este proceso fundamentalmente se aplican las técnicas, y método seleccionados. Las pruebas se realizan a nivel de pruebas de integración y sistema, debido a la fase del ciclo de vida en que se encuentra el software. Adicionalmente este proceso contiene las pruebas a la documentación, argumento importante a tener en cuenta al momento de evaluar la calidad de un producto software. No puede hablarse de un software con calidad si la documentación que lo patentiza no está a su nivel, por tanto hay que llevar con la misma rigurosidad ambas dimensiones, software-documentación.

1.8 Herramientas para automatizar las pruebas

Existen varias herramientas de automatización de las diferentes técnicas de pruebas. Entre ellas se pueden citar:

- **vTest:** Es una herramienta automática para pruebas funcionales y de regresión para aplicaciones web. Incorpora capacidades de registro, verificación, reproducción y reporte. No requiere conocimientos de programación. Para aquellos usuarios que desean escribir programas, éste usa JavaScript como la lengua de programación. Soporta tanto Microsoft Internet Explorer como Mozilla Firefox. [18]
- **AdventNet QEngine:** Es una herramienta para automatizar pruebas de funcionalidad, rendimiento, carga, tráfico de datos, servicios web, control del rendimiento de servidores y pruebas de regresión, para ejecutar pruebas desde la línea de comandos para todo tipo de

pruebas. QEngine funciona tanto en plataformas Windows como en Linux. Permite grabar y reproducir los exploradores Microsoft Internet Explorer, Firefox y Mozilla. [19]

- **AdventNet QEngine WebTest:** Posee una amplia gama de herramientas independientes de la plataforma, para pruebas de funcionalidad y de carga de la Red en aplicaciones web desarrolladas usando HTML, JSP, ASP, .NET, PHP, JavaScript/VBScript, e-commerce, etc. La herramienta de pruebas de Red ha sido desarrollada en Java, lo que facilita su portabilidad, soporte para múltiples plataformas (Windows, Linux, y Solaris) y para múltiples navegadores (FireFox, IE y Mozilla). [20]

- **Selenium IDE** es una herramienta para realización de pruebas de aplicaciones web, está orientado a la ejecución de pruebas funcionales a nivel de usuario directamente desde el navegador. Esta herramienta es muy útil para el desarrollo web donde se tienen que realizar montones de pruebas cada vez que se saca una versión nueva o se realizan modificaciones en un portal. Selenium automatiza el proceso de pruebas y permite ejecutar un conjunto de pruebas completo si es necesario o pruebas particulares. [21]

Las técnicas de pruebas seleccionadas no requieren una herramienta que las automatice, debido a que la complejidad de su ejecución es baja, y se pueden llevar a cabo manualmente. Para el caso de las pruebas de carga y estrés sí es imprescindible el uso de una herramienta capaz de automatizarlas, y lograr que se mida el rendimiento de un sistema de manera eficiente y eficaz. Resulta humanamente imposible simular de forma manual el comportamiento correspondiente a 200 personas que acceden concurrentemente a un sistema. Sería preciso emplear una cantidad insólita de recursos para lograr un comportamiento similar en el sistema y verificar que el desempeño es el esperado. Por estas razones se realizó una investigación de algunas herramientas encargadas de automatizar estas pruebas.

- **QALoad**

QALoad de la empresa Compuware, es una herramienta de pruebas para aquellas aplicaciones que estén sometidas a gran carga de usuarios. Los sistemas cliente/servidor de hoy en día todavía necesitan mejorar para permitir el uso concurrente a miles de usuarios. Las organizaciones necesitan desarrollar pruebas de carga repetibles y determinar el rendimiento real y límites potenciales del sistema. QALoad ayuda a alcanzar cargas que simulan el comportamiento real del negocio así como a validar que el sistema cumple aceptablemente con los niveles de servicio. [12]

Básicamente la herramienta proporciona:

- Pruebas escalables: Utilizando QALoad se puede emular la carga generada por cientos o miles de usuarios en la aplicación sin requerir la participación de los usuarios finales en sus equipos. Puede repetir fácilmente las pruebas de carga variando las configuraciones del sistema para alcanzar una aproximación óptima a los escenarios reales de uso. Con QALoad se configura un escenario de pruebas de carga para controlar las condiciones de las pruebas, crear los usuarios virtuales que se necesitan para simular la carga, iniciar y monitorizar las pruebas y se obtienen los informes de resultados.

- Fácil desarrollo de los scripts de prueba: QALoad ofrece módulos configurables que facilitan el desarrollo de los scripts de prueba. Estos módulos, llamados EasyScripts, permiten a los ingenieros de pruebas grabar el tráfico que genera la aplicación y convertirlo en un script de prueba. Los scripts de prueba resultantes reflejan el tráfico real generado por la aplicación y miden el tiempo empleado para completar las transacciones para garantizar que el sistema que sometido a las pruebas alcanza las especificaciones para las que ha sido construido.

- Análisis exhaustivo: Durante cada sesión de pruebas, QALoad ofrece las estadísticas de rendimiento en tiempo real, y permite a los ingenieros de prueba insertar puntos de control dentro de los scripts para identificar y revisar áreas específicas del rendimiento del sistema. QALoad muestra los resultados de las pruebas en una amplia variedad de informes y gráficas. Además, los datos de las sesiones de prueba se pueden exportar automáticamente a otros formatos para su revisión en detalle. [12]

Aunque pudiera ser ésta una herramienta candidata a seleccionar, constituye un software privativo, por tanto se precisa pagar para su adquisición, y por consiguiente también su documentación. Este hecho obstaculiza su elección.

- **LoadRunner**

LoadRunner es una herramienta para realizar pruebas de carga que permite prever el comportamiento y el rendimiento del sistema. Además, posibilita poner a prueba toda la infraestructura corporativa para identificar y aislar los posibles problemas mediante la simulación de la actividad de miles de usuarios. 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. [12]

A pesar de tener todas esas características favorables, su adquisición es sólo gratuita por 15 días, después de los cuales se debe pagar por su uso. Unido a esto la documentación está sujeta a las mismas restricciones, por lo que igual a QALoad no resulta favorable su selección.

- **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. [12]

Ésta 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. [12]

Esta herramienta maneja y automatiza el proceso de entrega con indicadores claves de rendimiento a tiempo real. Además reduce los riesgos de depuración de la aplicación. Cuenta con suficiente documentación en la web, a la que se tiene acceso gratis, en cambio, es desarrollada por empresas privativas fuera del marco del software libre, razón por la cual también queda fuera de la posible elección.

- **JMeter**

Es una herramienta de software libre que ofrece la posibilidad de realizar pruebas de carga sobre diferentes aspectos de una aplicación Web, inicialmente diseñada para pruebas de estrés en aplicaciones Web, en la actualidad, su arquitectura ha evolucionado no sólo para llevar a cabo pruebas en componentes habilitados en Internet (HTTP), sino además en bases de datos, programas en Perl y prácticamente cualquier otro medio.

Utilizar JMeter en aplicaciones web para la comprobación de los recursos del sistema, supone una mayor efectividad en el proceso y en la fiabilidad de los resultados. Como herramienta de prueba dispone de varios componentes que facilitan la elaboración de los escenarios de prueba con la ventaja de simular para cada uno de esos escenarios miles de usuarios. De esta manera se verifica el rendimiento del sistema mediante las pruebas de Carga y Estrés.

JMeter es fácilmente configurable y permite conocer los tiempos de respuesta experimentados por una aplicación cuando se tiene un número N de usuarios y el número real de transacciones procesadas por unidad de tiempo. Los resultados pueden ser visualizados en diferentes formatos, lo cual facilita las labores de análisis para el probador.

Una ventaja de JMeter es que permite realizar pruebas de carga sobre cada una de las capas que conforman la aplicación a probar. De esta manera, en un momento dado, es sencillo localizar el foco que está generando contención y está afectando los tiempos de respuesta de un caso de uso específico. [13]

1.8.1 Selección de la herramienta a utilizar

Algunos de los criterios analizados para seleccionar la herramienta apropiada para automatizar las pruebas de carga y estrés son:

- Funcionalidades: Prestaciones de la herramienta, elementos funcionales, respuesta a los requerimientos del sistema.
- Modo de adquisición: Si el software se obtiene por medio de una licencia de pago o es gratuito.
- Materiales de apoyo: Si existe documentación de la herramienta que ofrezca una guía para su utilización.
- Entorno en el que fue desarrollado: Si es un software a código abierto desarrollado en el ambiente de software libre o si es un software del cual no se puede obtener información de su

código o no se puede modificar y adecuar a los ambientes de trabajo del proyecto por ser un software propietario.

Del 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 proporcionando 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.

La utilización del JMeter supone un 95 % de tiempo menos para la realización de estas 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 y de la realización de un manual que permita a otros proyectos de gestión, identificar puntos de control para la elaboración de estas pruebas y la ejecución de las mismas, además de la recopilación y análisis de los errores encontrados.

Otro aspecto a favor de esta herramienta es su desarrollo en el marco del software libre, política que apoya la universidad y que por sus 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 única 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.

Conclusiones del capítulo

En el proceso de pruebas de liberación se ejecutan las pruebas de destrucción aleatoria, de seguridad y funcional, por ser ésta una técnica capaz de cubrir y evaluar las principales funcionalidades del software. Se aplicará el método de Caja Negra, por ser el más apropiado para probar la funcionalidad de un sistema. Para medir el rendimiento y la confiabilidad del software se aplican las pruebas de carga y estrés respectivamente, escogiéndose como herramienta de automatización JMeter, por sus particularidades de uso y ventajas.

CAPÍTULO 2

Diseño y aplicación de pruebas de liberación de software

Introducción

Este capítulo describe las características del SIGM en cuanto a módulos, casos de uso y criticidad de los mismos. Detalla el proceso de diseño y realización de las pruebas de liberación al Sistema Informático de Genética Médica partiendo de la estrategia elaborada para elaborar el mismo. Adicionalmente se describe detalladamente el proceso de automatización de las pruebas de carga y estrés con la herramienta JMeter.

2.1 Características del sistema

El Sistema Informático de Genética Médica está formado por la integración de 7 módulos o registros:

- Registro Cubano de Historias Clínicas Genéticas y Datos Primarios: Encarga do de gestionar los datos primarios del paciente y los de la historia clínica genética.
- Registro Cubano de Enfermedades Genéticas: Asociado a los datos de las enfermedades genéticas del paciente.
- Registro Cubano de Malformaciones Congénitas: Facultado para la gestión de la información asociada a los pacientes que sufran de malformaciones congénitas.
- Registro Cubano de Discapacitados: Gestiona la información relacionada con las discapacidades físicas y motoras de los pacientes.
- Registro Cubano de Retraso Mental: Se encarga de gestionar las discapacidades intelectuales o mentales.
- Registro Cubano de Gemelos: Gestiona la información referente al estudio de gemelos en el país y de los gemelos como tal.
- Registro Cubano de Teleconsulta Genética: Posibilita la discusión a distancia y en tiempo real de aquellos casos de los que no se tiene un diagnóstico certero.
- Administración: Aunque no se especifica directamente a las funcionalidades del sistema previamente establecidas por el cliente, constituye una funcionalidad implícita, ya que se ocupa de administrar los roles y permisos de los usuarios que tienen acceso a la aplicación.

En la siguiente tabla se desglosan por módulos los requerimientos funcionales con sus respectivos casos de uso asociados. En el caso del módulo de Administración, el equipo de desarrollo no definió casos de uso.

Tabla 2.1 Relación de requerimientos y casos de uso por módulos

Módulos	Requerimientos Funcionales	Casos de Uso	Prioridad
RECUDIS Registro Cubano de Discapacitados	RF1 Insertar datos complementarios de una persona discapacitada RF2 Modificar datos complementarios de una persona discapacitada RF3 Mostrar reportes estadísticos RF3.1 Listar la cantidad de discapacitados según su incapacidad RF3.2 Listar la cantidad de discapacitados según el sexo RF3.3 Listar la cantidad de discapacitados según su ocupación RF3.4 Listar la cantidad de discapacitados según el vínculo laboral RF3.5 Listar la cantidad de discapacitados según su capacidad laboral RF3.6 Determinar la cantidad de discapacitados según su amparo filial	Insertar Datos Complementarios	Crítico
		Modificar Datos Complementarios	Crítico
		Mostrar Reportes Estadísticos	Secundario

RECUEGEN Registro Cubano de Enfermedades Genéticas	RF1 Gestionar Reporte Estadístico RF2 Gestionar Cálculos totales de personas RF3 Gestionar Enfermedades Genéticas	Gestionar Enfermedades Genéticas	Crítico
		Gestionar Reportes de Cálculos Totales	Crítico
		Gestionar Reporte Estadístico	Crítico
RECUGEM Registro Cubano de Gemelos	RF1 Registrar nueva pareja de gemelos. RF2 Modificar datos de la pareja de gemelos. RF3 Mostrar instrumento de datos generales. RF4 Registrar datos de gemelos. RF5 Modificar datos de gemelos. RF6 Mostrar datos de gemelos. RF7 Buscar parejas de gemelos. RF8 Generar reportes estadísticos sobre datos generales: cantidad de parejas según la cigosidad. RF9 Generar reportes estadísticos sobre cantidad de parejas con gemelos fallecidos. RF10 Generar reportes estadísticos	Generar Reporte Estadístico	Secundario
		Gestionar Datos Complementarios del Gemelo	Crítico
		Gestionar Datos Primarios	Crítico
		Gestionar Pareja de Gemelos	Crítico

	sobre cantidad de parejas con más de dos gemelos. RF11 Generar reportes estadísticos sobre prevalencia de gemelos por cada mil habitantes a nivel provincial. RF12 Generar reportes avanzados para mostrar la cigosidad y la pareja a la que pertenecen los gemelos.		
RECUHC_DP Registro Cubano de Historias Clínicas Genéticas y Datos Primarios	RF1 Buscar Datos Primarios RF2 Insertar Datos Primarios RF3 Modificar Datos Primarios RF4 Registrar Datos Complementarios RF5 Modificar Datos Complementarios RF6 Registrar Consulta RF7 Actualizar Interconsulta RF8 Buscar General	Buscar General	Crítico
		Gestionar Datos Complementarios	Crítico
		Gestionar Datos Primarios	Crítico
		Registrar Consulta	Crítico
		Actualizar Interconsulta	Crítico
RECUMAC	RF1 Modificar datos del paciente RF2 Insertar datos complementarios del paciente RF3 Generar reportes	Codificar Malformación	Secundario
		Generar Reporte	Crítico

Registro Cubano de Malformaciones Congénitas	RF4 Codificar malformación RF5 Buscar pacientes sin codificar	Estadísticos	
		Gestionar Datos Primarios	Crítico
		Insertar Datos del Paciente	Crítico
		Modificar Datos del Paciente	Crítico
		Buscar Planilla	Secundario
RECURM Registro Cubano de Retraso Mental	RF1 Insertar datos complementarios del paciente RF2 Modificar los datos complementarios del paciente RF3 Generar reportes RF3.1 Determinar la cantidad de personas con retraso mental según el diagnóstico RF3.2 Determinar la cantidad de personas con retraso mental según la atención RF3.3 Mostrar las visitas a los	Gestionar Datos Primarios	Crítico
		Insertar Datos Complementarios Paciente	Crítico
		Modificar Datos Complementarios del Paciente	Secundario

	centros y el total		
		Generar Reportes	Secundario
RECUTL Registro Cubano de Teleconsulta Genética	RF1 Mostrar código de ética RF2 Introducir datos de la solicitud RF3 Buscar Historia Clínica RF4 Mostrar datos primarios de un paciente seleccionado RF5 Mostrar listado de las solicitudes en espera de aprobación organizadas por el nivel de urgencia y la fecha de arribo RF6 Aprobar solicitudes RF7 Denegar solicitudes RF8 Enviar mensaje notificando al solicitante que la solicitud fue denegada y su explicación RF9 Mostrar listado de casos a discutir planificados con su fecha y hora de realización RF10 Mostrar listado de los casos aprobados sin planificar organizados por el nivel de urgencia y la fecha de arribo	Administrar Caso en Discusión	Crítico
		Autorizar Solicitud	Secundario
		Citar Participante a la Discusión	Secundario
		Gestionar Caso a Discutir	Crítico
		Gestionar Datos Primarios	Crítico
		Gestionar Participante a la	Crítico

	RF11 Adicionar participantes RF12 Eliminar participantes RF13 Buscar participantes por los criterios: nombre, apellidos, municipio, provincia. RF14 Mostrar datos de los participantes RF15 Establecer fecha, hora de inicio y de fin de un caso a discutir RF16 Eliminar caso a discutir RF17 Modificar caso a discutir (fecha, hora de inicio y de fin, participantes) RF18 Enviar mensaje de citación a todos los participantes incluidos en la discusión de un caso RF19 Mostrar listado de casos a discutir planificados para el día de	Discusión	
		Notificar Negación de Solicitud	Secundario
		Participar en Discusión de un	Crítico
		Proponer Participantes a la Discusión	Secundario
		Solicitar Discusión de un Caso	Crítico
		Visualizar Informe	Secundario

	hoy RF20 Mostrar caso en discusión RF21 Mostrar datos del caso RF22 Mostrar sala de conversación RF23 Resolver caso RF24 Remitir caso RF25 Posponer caso RF26 Registrar informe del caso discutido RF27 Visualizar informe para ser aprobado RF28 Mostrar informes aprobados de los casos discutidos		

2.2 Estrategia de pruebas

Para llevar a cabo el proceso de pruebas de liberación se trazó una estrategia de manera que se organizara el procedimiento en función de realizar las pruebas a la aplicación y a la documentación. La secuencia de actividades a desarrollar queda reflejada en la siguiente figura:

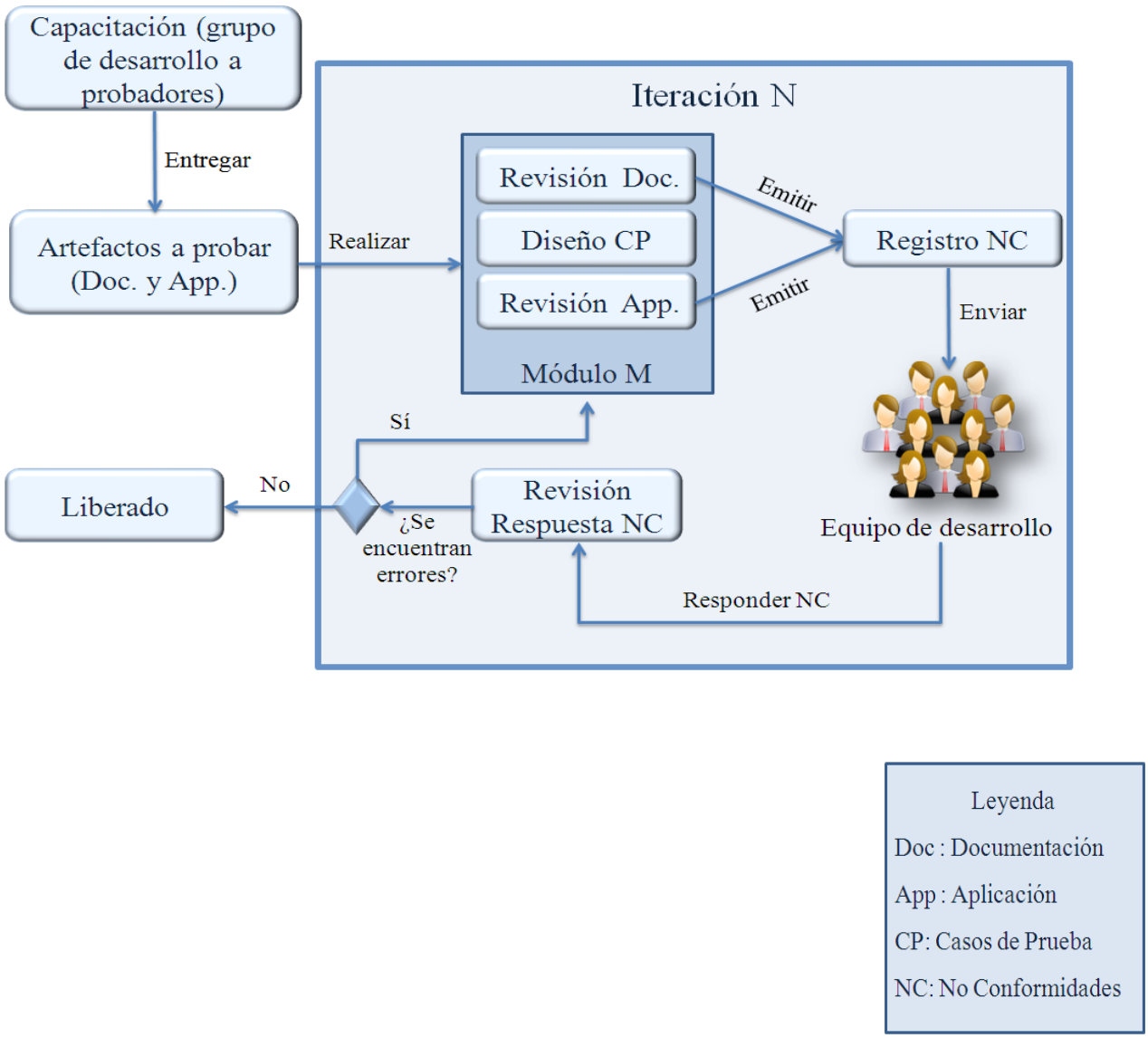


Ilustración 2.1 Actividades estrategia de pruebas

La estrategia trazada consta básicamente de los siguientes pasos:

- Elaboración del plan de pruebas.

- Definición de los niveles de prueba en los cuales se va a probar.
- Argumentación de las técnicas de prueba por cada nivel.
- Declaración del método de prueba a aplicar.
- Aplicación de listas de chequeo.
- Diseño de los casos de prueba y registro de no conformidades.
- Análisis de los resultados del proceso de pruebas de liberación.

Fuera del proceso de pruebas de liberación, a modo de enriquecer los criterios de evaluación del producto se añaden:

- Aplicación de pruebas de carga y estrés utilizando la herramienta JMeter.
- Análisis de los resultados de las pruebas automatizadas.

2.2.1 Plan de pruebas

El plan de pruebas constituye un artefacto de considerable importancia en el proceso de pruebas y es lo primero que se genera en este proceso, para planificar y trazar una estrategia. Proporciona el marco dentro del cual el equipo de probadores desarrolla las pruebas trabajando con los recursos y la planificación dada. De manera general suministra la siguiente información:

- La organización del equipo.
- La descripción de los requerimientos y los casos de uso.
- La definición de los elementos que se van a probar.
- Las técnicas y métodos de prueba.
- Una explicación del enfoque o estrategia que se usará.
- Los recursos y planificación necesarios.
- EL cronograma de actividades.
- Los resultados que se obtienen del proceso de prueba.

También se define el plan estratégico que se va a seguir en las pruebas. El plan de pruebas fue lo primero que se elaboró en el proceso de pruebas de liberación. Para mayor información del mismo remitirse al expediente de proyecto.

2.2.2 Niveles, técnicas y métodos de pruebas empleados

Los niveles en que se prueba no se pueden tratar de manera desligada del método y las técnicas empleadas, ya que ambas cosas están estrechamente relacionadas. Asimismo entre pruebas de integración y pruebas al sistema no hay diferencia notable, ya que en este último se verifica que todos los módulos trabajan como sistema sin error, sólo que con un alcance más amplio. De ahí que no se haga una distinción específica de las técnicas en uno u otro nivel, sino que se ven de manera complementada.

Las pruebas de integración se basaron en la verificación y validación de las funcionalidades de cada módulo por separado, para comprobar que éstos, como un conjunto de componentes integrados y combinados ejecutaban correctamente un caso de uso. Para estas pruebas se requirió de un mapa de integración proporcionado por el equipo de desarrollo, en el cual se reflejaba la manera en que los módulos se integran en el sistema según la dependencia jerárquica y la comunicación que existiera entre ellos.

Durante este nivel se aplicaron entre otras las pruebas funcionales, comprobando que se permitiera la navegación por toda la aplicación a medida que se probara la correspondencia entre las funciones implementadas con los requisitos del cliente. Para ello se diseñaron casos de pruebas basados en los casos de uso y a su vez en los requerimientos funcionales, comparando cada funcionalidad implementada con la descrita, para verificar hasta qué punto cumplía con las necesidades del cliente.

Se comprobó la correcta validación de los campos, verificando que cada cual aceptara exclusivamente los caracteres válidos. Por ejemplo en los campos donde su composición solo era de números se verificó si el sistema permitía entrar letras o algún otro carácter, dígase /, *, -, +, %, entre otros; asimismo para los campos donde sólo era necesario letras se verificó que no se insertaran números u otros caracteres no válidos. De esta forma se aplicó simultáneamente la técnica de Partición de Equivalencia de Caja Negra, con la comprobación de las variables válidas e inválidas. Cada defecto encontrado se fue registrando en la plantilla de no conformidades, argumentando cada no conformidad y clasificándola según su grado de importancia en significativa o no.

La seguridad del sistema se probó sobre el módulo de Administración. Por cada módulo, probando las funcionalidades de éste, se verificó que cada usuario por cada módulo tuviera los roles y privilegios asignados correctamente implementados.

Una de las formas en que se probó la confiabilidad fue con las pruebas aleatorias. Con la intención de bloquear la aplicación o provocar errores, se obviaba cualquier especificación del comportamiento del software y se proporcionan datos ilógicos y erróneos en los diferentes campos. De esta manera se podían encontrar las rutas de errores importantes y poner de manifiesto errores de programación del software

Es válido destacar que no sólo se midieron los parámetros de calidad mencionados, sino también todo aquel detalle que contribuyera a la presencia, y mejor aceptación del software por los usuarios finales. Elementos tales como errores ortográficos, la no alineación entre los diferentes campos, la estética de la interfaz de usuario, en fin, se trató de prever y erradicar todo aquello que podría constituir una no conformidad para el cliente.

2.2.3 Aplicación de listas de chequeo

La lista de chequeo es un documento que tiene un conjunto de parámetros a medir sobre un aspecto determinado, dígase documentación o aplicación. Es un instrumento de medición y evaluación que consiste básicamente en un formulario de preguntas referentes al atributo de calidad que se está probando y de las características del documento en el caso de la documentación. Cada pregunta tiene asociada una evaluación en una escala que da una medida del grado de cumplimiento y disponibilidad de la propiedad evaluada, de esta manera se determina la evaluación del elemento probado.

En materia de documentación se utilizó una lista de chequeo por cada documento revisado, dígase Manual de usuario, Especificación de requerimientos, Descripción de los casos de uso y Diccionario de Datos, de cada módulo pertenecientes al flujo de Requerimientos, así como la planilla de Roles y Permisos.

Se tuvo en cuenta aspectos significativos como:
Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.
Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.
Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.

Para evaluar estos criterios se plantearon una serie de interrogantes que contribuyeran a la determinación de las no conformidades. Algunas de ellas se presentan a continuación:

En el documento de Especificación de requerimiento:

¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?

¿Debería especificarse algún requisito con más detalle?

¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no como el sistema debe hacerlo?

¿Se han identificado los requerimientos de software y de hardware?

¿Los requerimientos de soporte y usabilidad se han identificado?

¿Se puede verificar cada requisito? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requerimiento, ejemplo la especificación del caso de uso).

¿Se han enumerado los requisitos incluso los que se derivan de otros requisitos?

¿Se han identificado errores ortográficos?

¿Se entiende claramente lo que se ha especificado en el documento?

En el documento de Modelo de CUS:

¿Cada caso de uso registra claramente lo que el sistema debe hacer?

¿Están clasificados los casos de uso que definen la arquitectura básica del sistema?

¿Se ha descrito con precisión todas las alternativas o excepciones?

¿Están clasificados los casos de uso que no son claves para la arquitectura?

¿Está en infinitivo y refleja de manera clara el objetivo del usuario sobre el sistema?

¿El nombre del caso de uso es único?

¿El nombre del caso de uso es intuitivo?

¿El resumen dice cómo se inicia, cómo termina y las operaciones principales que realiza el caso de uso?

¿Se especifica la complejidad del caso de uso?

¿Está descrito el caso de uso en presente?

¿Los flujos alternativos se nombran con el número del paso que lo generó en el flujo básico?

¿En la sección flujos alternativos se describen todas las excepciones que existan por muy evidentes que parezcan?

¿Se han identificado errores ortográficos?

¿Se entiende claramente lo que se ha especificado en el documento?

En el documento Diccionario de datos:

¿Cada descripción del elemento de datos es clara y detallada?

¿Han sido registrados todos los detalles adicionales relacionados al flujo de datos del sistema?

¿Ha especificado si el elemento de datos contiene valor numérico, caracteres o alfabético?

¿Ha especificado el máximo de caracteres o dígitos que puede tener un elemento de datos?

¿Ha registrado qué tipo de valor específico debe tener el elemento de datos?

¿Se han identificado errores ortográficos?

¿Se entiende claramente lo que se ha especificado en el documento?

En el documento Manual de usuario:

¿Se especifica detalladamente cada interfaz del flujo de eventos a probar?

¿Se explicitan todos los campos y funcionalidades de éstos por interfaz?

¿Corresponde la verdadera interfaz de usuario con la señalada en el documento?

¿Se han identificado errores ortográficos?

¿Se entiende claramente lo que se ha especificado en el documento?

Para probar la aplicación las listas de chequeo verifican el cumplimiento de los atributos de calidad enmarcándose en la verificación de la confiabilidad, seguridad, portabilidad, usabilidad y eficiencia. Para este caso las preguntas que se tuvieron en cuenta por cada atributo se muestran a continuación.

Usabilidad

¿Proporciona el sistema un soporte apropiado a los usuarios más novatos?

¿Resulta fácil instalar el software?

¿Se entienden la interfaz y su contenido?

¿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?

¿Es fácil de utilizar el sistema en la realización de tareas?

¿Está diseñada la interfaz para facilitar la realización eficiente de las tareas de la mejor forma posible?

¿Actúa el sistema como corrector de errores?

¿Actúa el sistema en la prevención de errores?

¿Actúa el sistema en la información de los errores?

¿Se permite la utilización del ratón o el teclado?
¿Permite al usuario interrumpir su tarea y continuar más tarde?
¿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?
¿Proporciona funciones deshacer, rehacer?
¿Se presenta al usuario la información que sólo necesita?

Seguridad

¿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?
¿Toda la privacidad, la libertad de información, sensibilidad, y consideraciones de la clasificación fueron identificadas, resueltas, y establecidas?

Confiabilidad

¿Se recupera el software ante fallas?
¿La recuperación ante fallas se realiza en el tiempo requerido para la aplicación?

Portabilidad

¿Existe independencia del ambiente de software (sistema operativo, lenguaje de programación)?

2.2.4 Diseño de casos de prueba y registro de no conformidades

Según los lineamientos de calidad, un producto software debe estar bien elaborado y documentado, uno de los propósitos con que se realiza esta documentación es de explicar cada una de las funcionalidades que se implementan en el sistema.

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previamente a realizar las pruebas funcionales a la aplicación. Se parte de la descripción de los casos de usos del sistema, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión, además quedan plasmadas las revisiones realizadas al caso de prueba; así como un registro de todo aquello que no corresponde a la calidad del software. Existen casos de pruebas para los diferentes métodos: Caja Negra y Caja Blanca. En el proceso de pruebas en cuestión se hace uso de los casos

de prueba de Caja Negra por ser éste el método empleado en la liberación del producto software, de ahí el motivo por el que se diseña un caso de prueba por caso de uso del sistema.

Partiendo de la descripción de los casos de usos del sistema, como apoyo para las revisiones, se diseñó un caso de prueba asociado a cada caso de uso (ver expediente de proyecto). Para detallar el caso de uso se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y a su vez estas en escenarios, para hacer más fructífera la ejecución de las pruebas. Esta tabla contiene los campos:

- Nombre de la sección: Se especifica el nombre de la sección [SC 1: Nombre de la sección].
- Escenarios de la sección: Se especifican los escenarios de cada sección [EC 1.1: Nombre del Escenario].
- Descripción de la funcionalidad: Se describe brevemente la funcionalidad del escenario.

El siguiente es un ejemplo donde se detalla el caso de uso Insertar Datos Complementarios del módulo RECUDIS.

Tabla 2.2 Secciones a probar en el Caso de Uso

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Insertar Datos Complementarios.	EC 1.1: Insertar Datos Complementarios.	En este escenario el defectólogo selecciona el individuo que desea insertar, luego introducen los datos correspondientes a la discapacidad.
	EC1.2: Insertar datos Complementarios con campos vacios.	Es la misma operación que se realiza en el EC 1.1, en caso de que no se llenen los campos requeridos.

A partir de esta descripción se detallan las variables que se encontraban en todas las interfaces asociadas al caso de uso que se le estaba diseñando el caso de prueba. Esta descripción está compuesta por campos tales como:

- No: se enumeró todos los campos o variable, descrito en el caso de uso.
- Nombre de campo: se especificó el nombre del campo de entrada.

- Clasificación: se especificó la clasificación según el componente de diseño utilizado [ejemplo: campo de texto, lista desplegable o combo box, entre otros].
- Puede ser nulo: se especificó si el campo puede ser nulo o no, para ello solo se puso Sí o No.
- Descripción: se describieron brevemente los datos que debían introducirse.

En la siguiente tabla se muestra un ejemplo de la descripción de variables del caso de uso Insertar Datos Complementarios del módulo RECUDIS.

Tabla 2.3 Descripción de variables

No.	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Escolaridad	Lista de selección	No	Se tiene que escoger el nivel escolar de la persona discapacitada.
2	Tipo de Discapacidad	Lista de selección	No	Se escogen las discapacidades que presenta la persona.
3	Evaluación Funcional	Lista de selección	No	Si la persona es capaz de valerse por sí sola.
4	Condiciones de la Vivienda	Lista de selección	No	Se escoge la situación en que se encuentra la vivienda.
5	Higiene de la Vivienda	Lista de selección	No	Se escoge la situación higiénica en que se encuentra la casa de la persona discapacitada.
6	Número de personas en el dormitorio del discapacitado	Entero	No	La cantidad de personas que duermen en el mismo cuarto de la persona discapacitada.
7	Accesibilidad a la	Lista de	No	Se escoge como es la situación

	transportación	selección		de la persona discapacitada a acceder a la transportación.
8	Composición del núcleo familiar	Lista de selección	No	Se seleccionan las personas que conviven con la persona discapacitada.
9	Amparo Filial	Lista de selección	No	Se selecciona si a la persona discapacitada se le brinda apoyo familiar.
10	Ingreso total del núcleo	Entero.	No	La cantidad total de dinero que entra a la casa de la persona discapacitada mensualmente.
11	Ingreso Percápita	Entero	No	Una aproximación del promedio del dinero que entra por personas a la familia.
12	No. de convivientes en el núcleo	Entero	No	La cantidad de personas que conviven en la casa con la persona discapacitada.
13	Capacidad laboral	Selección	No	Si la persona discapacitada está apta para trabajar o no.
14	Salario	Selección	No	Si la persona discapacitada tiene o no un salario mensual.
15	Ocupación del discapacitado	Selección	No	Seleccionar la ocupación laboral que se encuentra haciendo actualmente.

16	Vínculo Laboral	Selección	No	Seleccionar que labor se encuentra haciendo actualmente.
17	Miembros del núcleo que consumen alcohol	Selección	No	Se selecciona de qué forma consumen alcohol los integrantes del núcleo familiar.

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos validos e inválidos se identificó el empleo de la técnica de partición de equivalencia. Esta matriz de datos contiene los siguientes aspectos:

- Id del escenario: Se especifica el id del escenario [EC1, EC2,…,ECn]
- Escenario: Se especifica el nombre del escenario.
- Variables [1, 2,…, n]: Se especifica el nombre de la variable, o el número según la tabla de descripción de variables y en su celda correspondiente se indicó el valor del dato [V (Válido), I (Inválido), N/A (No Aplica)].
- Respuesta del sistema: Se escribe el resultado que se esperaba al realizar la prueba.
- Resultado de la prueba: Se escribe el resultado que se obtuvo al realizar la prueba.
- Flujo central: Se expone el flujo central del caso de uso al que se le estaba diseñando el caso de prueba.

A continuación un ejemplo de la matriz de datos del caso de uso Insertar Datos Complementarios, de la SC 1 Insertar Datos Complementarios:

5.1. SC 1 Insertar Datos Complementarios

Id del escenario	Escenario	Variables (Enumeradas según Tabla 1.1 Descripción de Variables)																	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17			
EC 1.1	Insertar Datos Complementarios.	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	El sistema registra los datos de la persona discapacitada. El sistema emite un mensaje mostrando que se han insertado los datos correctamente	Satisfactorio	1.-El defectólogo selecciona la opción de insertar los datos complementarios. 2.-Se llama al Caso de uso incluido "**Gestionar Datos Primarios**" para buscar a la persona a la cual se le van a insertar los datos complementarios. 3.- El defectólogo selecciona a la persona con discapacidad al cual le va a llenar los campos complementarios. 4.- El sistema le brinda una interfaz que muestra los criterios para la inserción de los datos. 5.- El defectólogo registra los datos complementarios del discapacitado 6.-El sistema verifica que no se haya quedado ningún campo obligatorio vacío. 7.- El sistema registra los datos de la persona discapacitada. 8.-El sistema emite un mensaje mostrando que se han insertado los datos correctamente.
EC 1.2	Insertar datos Complementarios con campos vacios.	I	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	El sistema emite un mensaje para llenar los campos que son obligatorios.	Satisfactorio	6.1 Se emite un mensaje para llenar los campos que son obligatorios.
		V	I	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V			
		V	V	I	V	V	V	V	V	V	V	V	V	V	V	V	V	V			
		V	V	V	I	V	V	V	V	V	V	V	V	V	V	V	V	V			
		V	V	V	V	I	V	V	V	V	V	V	V	V	V	V	V	V			
		V	V	V	V	V	I	V	V	V	V	V	V	V	V	V	V	V			
		V	V	V	V	V	V	I	V	V	V	V	V	V	V	V	V	V			

Ilustración 2.2 Matriz de Datos

Los resultados de las pruebas que no fueron satisfactorios pasaron a ser no conformidades y se emitieron en el registro de defectos y dificultades detectados que se encuentra en la parte final de cada diseño de caso de prueba con los siguientes campos:

- Elemento: Se especifica el nombre del elemento.
- No: Se especifica el número de la no conformidad.
- No conformidad: Se describe la no conformidad.
- Aspecto correspondiente: Se especifica el aspecto correspondiente a la no conformidad.
- Etapa de detección: Se especifica la etapa de detección del error.
- Significativa: Se pone una (X), en caso de que la no conformidad estuviera clasificada como significativa.
- No significativa: Se pone una (X), en caso de que la no conformidad estuviera clasificada como no significativa.
- Recomendación: Se pone una (X), en caso de que la no conformidad solo fuera una recomendación.
- Estado NC: Se coloca el estado de la no conformidad y la fecha, cada vez que se revisa se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó [RA: Resuelta, PD: Pendiente, NP: No Procede].
- Respuesta del equipo de desarrollo: Esta columna se comienza a llenar a partir de la segunda iteración, y es responsabilidad del equipo de desarrollo, quien especifica si se ha resuelto o no la no conformidad y en caso de no proceder se explica la causa.

En la siguiente ilustración se muestra un ejemplo de algunas no conformidades detectadas durante las pruebas al caso de uso Insertar Datos Complementarios del módulo RECUDIS.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	Al insertar los Datos Complementarios de un discapacitado, en la sección: Tipo de discapacidad y Evaluación funcional, en el Tipo de Discapacidad Auditiva, la palabra oído, está sin tilde.	http://10.7.9.200:580/dis/insertar/id/514	Pruebas de funcionalidad y confiabilidad		X		13/10/2008 PD 11/11/2008 RA	Se le puso la tilde a "oído"
Aplicación	2	Al insertar los Datos Complementarios de un discapacitado, en la sección: Datos Primarios, aparece la palabrea Area sin tilde.	http://10.7.9.200:580/dis/insertar/id/514	Pruebas de funcionalidad y confiabilidad		X		13/10/2008 PD 11/11/2008 RA	Se le puso la tilde a la palabra "área"
Aplicación	3	Al insertar los Datos Complementarios de un discapacitado, en la sección: Datos particulares del Discapacitado, al seleccionar la opción No, el verbo está, no tiene tilde.	http://10.7.9.200:580/dis/insertar/id/514	Pruebas de funcionalidad y confiabilidad	X			13/10/2008 PD 11/11/2008 RA	Se le puso la tilde a la palabra está
Aplicación	4	En la sección: Datos particulares del Discapacitado, al seleccionar la opción Sí, aparece por defecto el componente que permite especificar Otra asociación, sin antes haber marcado la opción Otra. **Ver anexo 1.**	http://10.7.9.200:580/dis/insertar/id/514	Pruebas de funcionalidad y confiabilidad		X		13/10/2008 PD 11/11/2008 RA	

Ilustración 2.3 Registro de defectos y dificultades detectados

La plantilla de No Conformidades recoge además los errores que son detectados durante la revisión de la documentación del sistema. Se elabora un documento por cada revisión que se haga y se controlan a través de versiones según se vayan eliminando los errores, hasta que finalmente se hayan erradicado todos los defectos que posea el elemento que se prueba. Además de estar inmerso en la planilla de diseño de casos de prueba, estas no conformidades se van registrando en un documento aparte para luego enviarlo al equipo de desarrolladores. De igual manera, aún trabajando paralelamente con los casos de prueba, el documento emitido a los desarrolladores es independiente, porque los casos de prueba son para consumo y único uso de los probadores.

2.3 Aplicación de Pruebas de Carga y Estrés utilizando la herramienta JMeter

Con las pruebas que tradicionalmente se tienen estipuladas no se abarcan todos los riesgos de calidad de software; de ahí que el equipo de pruebas considere importante la inclusión de estas pruebas en el proceso de pruebas de liberación establecido en la universidad.

Objetivo

El objetivo central de las pruebas de carga y estrés es esencialmente obtener un índice de resultados de comportamiento del sistema en determinadas condiciones, o sea, al realizar las pruebas de estrés y carga a la aplicación se obtienen resultados de un comportamiento que puede ser muy cercano a un comportamiento real, con ello se espera realizar un análisis en el equipo de desarrollo para en caso de ser resultados poco favorables realizar acciones que mitiguen las deficiencias. Teniendo estos resultados se puede prever errores futuros y respuestas del sistema ante cantidades específicas de usuarios y ante altos niveles de concurrencia y grandes volúmenes de información. [12]

Especificaciones de rendimiento

Al realizar las pruebas de carga y estrés es necesario tener declarados explícitamente por parte del equipo de desarrollo las especificaciones de rendimiento, dígase aquellos requerimientos propios del sistema que describen las características de rendimiento que se supone deba tener la aplicación en un ambiente real de ejecución. Esta práctica en la actualidad se omite por muchos desarrolladores porque no la consideran tan importante como los requerimientos funcionales y no funcionales citados por el cliente, sin embargo según la experiencia acumulada en las pruebas de software, se puede afirmar que estos requisitos tienen tanta trascendencia como los usualmente declarados en el contrato con los clientes. Un software en el que no se han definido las especificaciones de rendimiento, tiene gran probabilidad de que colapse ante una determinada carga de trabajo no preconcebida, un número de peticiones tal que no se haya tenido en cuenta durante el desarrollo, u otra situación no prevista que provoque un mal funcionamiento del software. Con ello además se comprueba que la base de datos del sistema esté en condiciones lo suficientemente confiables para aceptar el volumen de información que se precise.

Siempre es útil disponer de una estimación del pico de número de usuarios que se espera que utilicen el sistema en las horas punta. Si puede ser, también una estimación del máximo tiempo de respuesta permitido en el percentil 95, para que la configuración de la ejecución de las pruebas se ajuste a estas especificaciones.

Es válido señalar que los resultados arrojados por la herramienta tienen relación directa con los requisitos de rendimiento, así, con ambos datos se puede hacer una evaluación del software en términos de desempeño y con el tiempo justo, resolver las deficiencias que se encuentren en los resultados de las pruebas.

Para enunciar las citadas especificaciones, serían de gran utilidad las siguientes preguntas:

- ¿Cuál es el alcance, en detalle, de la prueba de rendimiento? ¿Qué subsistemas, interfaces, componentes, etc. están dentro y fuera del ámbito de ejecución de esta prueba?
- Para las interfaces de usuario involucradas, ¿Cual es el número de usuarios concurrentes que se esperan para cada uno, especificando picos y medias?
- ¿Cuál es la estructura objetivo del sistema hardware, especificando todos los servidores de red y configuraciones de dispositivo?
- ¿Cuál es la distribución del volumen de trabajo de la aplicación para cada componente?
- ¿Cuál es la distribución del trabajo del sistema? Las cargas de trabajo múltiples pueden ser simuladas en una sola prueba de eficacia (por ejemplo: 30% del volumen de trabajo para A, 20% del volumen de trabajo para B, 50% del volumen de trabajo para C).
- ¿Cuáles son los requisitos de tiempo para cada uno y para todos los procesos por lotes, especificando picos y medias?
- Identificar los criterios de aceptación de rendimiento. Determinar el tiempo de respuesta, el rendimiento, la utilización de los recursos y los objetivos y limitaciones. En general, el tiempo de respuesta concierne al usuario, el rendimiento al negocio, y la utilización de los recursos al sistema.

Técnica y Método

Se empleó la técnica de Pruebas Funcionales previamente explicada conjuntamente con el método de Caja Negra, por ser una prueba que necesita la entrada de datos y la validación de ellos contra resultados esperados.

2.3.1 Descripción de la ejecución de las pruebas

Para la ejecución de las pruebas se hizo una selección modesta de los casos de uso, de manera que se tuviera una representación de todas las funcionalidades del sistema distribuidas por todos sus módulos (Ver Tabla 2.4). Además resaltar que no es necesario probar cada caso de uso para arrojar los resultados que se analizan con vista a evaluar el desempeño del software.

En dependencia del sistema a probar se ejecutan script de pruebas simples o complejos. En una aplicación web donde solo se modele la navegación de los usuarios se pueden realizar los scripts de prueba simples o pruebas de navegación. En sistemas más complejos donde se manejen grandes volúmenes de datos y se gestione información, se recomienda el uso de script de pruebas complejas, donde se modela la interacción del usuario con la aplicación mediante la entrada y salida de datos.
Para ambos casos el proceso de automatización de las pruebas consta de 4 etapas fundamentales.

- Grabación de los escenarios de prueba.
- Confección de los test.
- Ejecución de las pruebas y recolección de los resultados.
- Análisis de los resultados.

Se considera que las pruebas simples no se corresponden con las características propias del SIGM, por cuanto exclusivamente se ejecutarán las pruebas complejas.

Tabla 2.4 Casos de Uso a probar en JMeter

Módulo	Caso de Uso Crítico
RECUHC_DP	CU_Gestionar Datos Primarios CU_Registrar Consulta
RECUEGEN	CU_Gestionar Enfermedades Genéticas
RECUMAC	CU_Modificar Datos Complementarios
RECUDIS	CU_Mostrar Reportes Estadísticos
RECURM	CU_Insertar Datos Complementarios
RECUGEM	CU_Gestionar Pareja de Gemelos
RECUTL	CU_Solicitar Discusión de un Caso

Grabación de los escenarios

Previamente a grabar los escenarios, se creó el Grupo de Hilos (Usuarios Concurrentes), que representa el número de usuarios que va a acceder concurrentemente al sistema. Tributando a los casos de uso correspondientes a cada módulo que integra el sistema, se realizaron 7 grabaciones. Para la realización de estas pruebas, inicialmente se debió configurar el navegador, especificando la dirección y el nuevo puerto a utilizar para la grabación de los escenarios, dicho puerto se corresponde

con el del servidor de JMeter. Una vez configurado el navegador a través de la opción Arrancar, se procedió a navegar por la aplicación según los casos de uso seleccionados, siguiendo el flujo básico descrito en los casos de prueba y probando simultáneamente la funcionalidad. De esta manera los escenarios quedaron grabados automáticamente. Con esta grabación, quedaron indicadas todas las peticiones que se realizan normalmente por los usuarios en un ambiente real.

Confección de los test

Posterior a las grabaciones se procedió a personalizar el entorno de cada grabación realizada, añadiendo elementos significativos para el monitoreo y seguimiento de resultados; dígase Informe Agregado, Árbol de Resultados, Cantidad de Hilos, la organización de las peticiones en diferentes controladores y la especificación de la cantidad de veces que se necesita la ejecución de estas peticiones durante la prueba (ciclos). En un primer momento se realizaron 3 iteraciones con el servidor de aplicación corriendo en el sistema operativo Linux, probando todos los escenarios grabados. Para una primera iteración se definieron 100 usuarios, donde se especifica además el número de ciclos y el período de subida *(*representa el tiempo entre cada ciclo). De esta manera la prueba queda configurada para simular un total de 100 usuarios en un estimado de tiempo de un segundo, para un sólo ciclo. En una segunda iteración se simularon 200 usuarios en un período de subida de 1 segundo. Finalmente, se procedió a elaborar una última iteración para un total de 300 usuarios, registrándose en cada iteración los resultados según la cantidad de usuarios a que se sometió el sistema.
Lo anteriormente explicado se ejecutó de igual manera pero en este caso con el servidor de aplicación sobre el sistema operativo Windows, a modo de comparar el comportamiento del software en ambos sistemas operativos.

Ejecución de las pruebas y recolección de los resultados

Para ejecutar las pruebas se seleccionó nuevamente la opción Arrancar y la herramienta comienza a simular el uso de la aplicación para las especificaciones mencionadas, antes se debe garantizar previamente el buen funcionamiento de la aplicación. Las respuestas de las diferentes peticiones se van guardando en el Informe Agregado y el Árbol de Resultados.

Análisis de los resultados

Los resultados recopilados de las pruebas fueron analizados, destacándose los aspectos más importantes que pueden interferir en gran medida en el rendimiento del sistema, tomándose decisiones por parte del equipo de desarrollo para mejorar esos resultados. La base de los seguimientos está

centrada en la reincidencia de los elementos encontrados, los cuales deben ser depurados antes de la siguiente iteración de pruebas.

Descripción de los aspectos mostrados en la tabla de la Plantilla de Reporte de los Resultados:

Muestras: Cantidad de páginas (Hilos) que simulan la cantidad de usuarios que están interactuando con el sistema desde la misma URL.
Media: Media de tiempo en milisegundos en que las páginas que se carga ron de manera satisfactoria.
Mediana: Tiempo promedio que han tardado en cargarse las páginas.
Min: Tiempo mínimo que ha demorado en cargarse una página.
Max: Tiempo Máximo que ha tardado en cargarse una página.
Línea 90 %: 90 por ciento del tiempo en el que las páginas que se carga ron de manera satisfactoria.
%Error: Por ciento de error de las páginas que no se llegaron a carga r de manera satisfactoria.
Kb/Seg: Velocidad de carga de las páginas.
Rendimiento: Representa el número de muestras por unidad de tiempo.

Un análisis eficiente de los resultados requiere además considerar el tiempo de demora de cada transacción; analizar por qué unas demoran más que otras y si estos tiempos pueden ser mejorados. En caso de fallar alguna de las peticiones se debe analizar sus causas. Además conviene verificar que las peticiones que se estén realizando al servidor sean realmente necesarias y que no sean redundantes.
Es válido señalar que el SIGM no tiene entre los requerimientos pactados con el cliente, las especificaciones de rendimiento para hacer la comparación con los índices que arroja JMeter, por lo que la evaluación final del comportamiento del software no es precisa en cuanto al desempeño que pudiera tener en condiciones muy cercanas al ambiente real.

Conclusiones del capítulo

Durante el desarrollo del presente capítulo se describió el proceso de pruebas de liberación del SIGM, basado en la estrategia elaborada. Se realizaron pruebas a la documentación y a la aplicación, utilizando listas de chequeo; para ello se aplicaron las técnicas y método en los niveles seleccionados. Con la intención de una mejor organización de la ejecución de las pruebas se diseñaron los casos de prueba. Los resultados arrojados de todas las pruebas realizadas (incluso a la documentación del sistema) se archivaron en la plantilla de no conformidades, para darle seguimiento y garantizar que se

fueran eliminando a medida de avanzar en el proceso de pruebas. Para medir el rendimiento del sistema se realizaron las pruebas de Carga y Estrés automatizándolas con la herramienta JMeter.

CAPÍTULO 3

Análisis de los resultados

Introducción

Todo proceso de pruebas requiere de un análisis final de los resultados obtenidos, de forma tal que se proporcione una evaluación del producto que se prueba teniendo en cuenta el seguimiento que se ha ido registrando de los defectos y fallos del sistema durante todo el proceso. Por cuanto en este capítulo se procura analizar las no conformidades encontradas según el elemento probado y técnica empleada. Se hace una valoración del producto de acuerdo a parámetros de calidad como la seguridad, portabilidad, usabilidad y confiabilidad, basándose en los resultados obtenidos.

3.1 Análisis de las No Conformidades de la documentación

Una vez registradas todas las no conformidades encontradas (ver expediente de proyecto) durante la revisión de la documentación es necesario contabilizar a modo de resumen el total de no conformidades por módulo, distinguiendo de ellas cuáles son de importancia alta y cuáles no. Las no conformidades significativas de la documentación se clasifican según los siguientes criterios:

- Ortografía
- Redacción
- Correspondencia con otra documentación
- Formato
- Error técnico

A continuación se muestra un ejemplo de una muestra de no conformidades del diccionario de datos del módulo RECUMAC:

Tabla 3.1 No Conformidades Diccionario de Datos RECUMAC

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa
Documentación	1	Los paréntesis angulares son para indicar	Portada del Doc. Diccionario de Datos, Control de	Revisión de la Documentación		X

		algo que no debe faltar, por tanto al poner el nombre del proyecto o del módulo al que se le está haciendo la documentación se deben quitar.	Versiones, Tabla de Contenido (revisar todo el documento).			
Documentación	2	La palabra *dio* no se acentúa.	Pág. 6 Tabla Datos de la Madre campo Hospital.	Revisión de la Documentación	X	
Documentación	3	Cambio de preposición *de* por la forma pronominal *se.*	Pág. 9, 14 campo Malformación.	Revisión de la Documentación	X	

En la siguiente tabla se expone un resumen de los resultados obtenidos durante la revisión de la documentación.

Tabla 3.2 Resumen de NC de la documentación

	Total de No Conformidades	# de No Conformidades Significativas	# de No Conformidades No Significativas
RECUDIS	**38**	**31**	**7**

RECUEGEN	**30**	**30**	**-**
RECUGEM	**30**	**26**	**4**
RECUHC_DP	**21**	**13**	**8**
RECUMAC	**27**	**21**	**6**
RECURM	**7**	**5**	**2**
RECUTL	**2**	**2**	**-**
ADMON	**2**	**2**	**-**
TOTAL	**157**	**130**	**27**

En la tabla anterior se refleja un balance de las no conformidades contenidas en todas las iteraciones. En el caso de los módulos RECUMAC y RECUHC_DP tuvieron tres iteraciones, ADMON una y el resto dos. A medida que se iban erradicando los errores se iba liberando la documentación del módulo que se estaba revisando, hasta quedar una documentación libre de defectos y lista para ser entregada al usuario final.

3.2 Análisis de las No Conformidades de la aplicación

Para clasificar las no conformidades en significativas de la aplicación se tuvieron en cuenta los siguientes criterios:

- Ortografía
- Funcionalidad
- Validación
- Excepciones
- Correspondencia con documentación
- Opciones que no funcionan

Las no significativas son aquellas que tenían una importancia alta ni afectaban la funcionalidad del sistema.

Además se valoró la ocurrencia de alguna función inutilizable que causara un término anormal o una falla general, o no se correspondiera con los resultados esperados. En el caso de las no significativas se comprobó el tiempo de respuesta de una función determinada, o que ésta no se ajustara a las normas y convenciones establecidas. También están incluidas en esta clasificación, los casos en los

que hubiera palabras con errores ortográficos, mensajes de error inconsistentes, y otros criterios de estética y estímulo visual que le restaran calidad al sistema.

Un ejemplo de las no conformidades de la aplicación para el módulo RECUMAC, de las funcionalidades insertar y modificar datos complementarios se muestra a continuación:

Tabla 3.3 No Conformidades aplicación módulo RECUMAC

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa
Aplicación	1	Al insertar los datos complementarios de un paciente, el campo *Nombre* en *Datos iniciales* del paciente acepta cualquier tipo de caracteres. Además de que es el único campo editable en esta sección.	http://10.7.9.200:5801/mc/agregarmf/id/537	Pruebas de funcionalidad y confiabilidad	X	
Aplicación	2	Al insertar datos complementarios, en la lista desplegable del campo	http://10.7.9.200:5801/mc/agregarmf/id/537	Pruebas de funcionalidad y confiabilidad		X

		Tipo de parto en *Datos de la madre* no se muestran las opciones completamente.				
Aplicación	3	Al insertar datos complementarios cuando se presiona el botón Aceptar, me la aplicación mostró un mensaje de error: *No puede seleccionar pacientes que no sean de su provincia,* cuando ese paciente no podía ser de otra provincia que no fuera Pinar porque la búsqueda	http://10.7.9.200:5801/gm/error/iderr/8	Pruebas de funcionalidad y confiabilidad	X	

		inicial es restringida sólo para esta provincia.				

Un resumen de los resultados obtenidos durante las pruebas a la aplicación se representa en la siguiente tabla.

Tabla 3.4 Resumen de las No Conformidades de la aplicación

	Total de No Conformidades	# de No Conformidades Significativas	# de No Conformidades No Significativas
RECUDIS	**20**	**14**	**6**
RECUEGEN	**29**	**24**	**5**
RECUGEM	**7**	**5**	**2**
RECUHC_DP	**44**	**32**	**12**
RECUMAC	**17**	**15**	**2**
RECURM	**23**	**17**	**6**
RECUTL	**17**	**17**	**-**
Integración SIGM	**3**	**3**	**-**
TOTAL	**160**	**127**	**33**

Tiene lugar acotar que en el caso de RECUMAC con seis iteraciones fue el módulo que más tuvo, seguido de RECUEGEN con cuatro, RECUHC_DP con tres y los restantes con dos. Luego de realizar las pruebas a los módulos individualmente, se procedió a integrar todas las funcionalidades según el mapa de integración proporcionado por el equipo de desarrollo. A medida que se iba avanzando en las iteraciones disminuían las no conformidades, hasta finalmente quedar libre de errores la aplicación y lista para ser entregada al cliente. (Ver expediente de proyecto)

3.3 Análisis de los resultados de las pruebas de Carga y Estrés con la herramienta JMeter

El último paso a realizar durante la ejecución de las pruebas en JMeter es el análisis de los resultados arrojados en el Informe Agregado que se genera en la herramienta a medida que se van probando los escenarios previamente grabados. Antes de comenzar con el análisis cuantitativo de los índices obtenidos es elemental declarar las condiciones de hardware en que se desarrollaron las pruebas, así como los requerimientos no funcionales de hardware y software que precisan JMeter y el sistema en cuestión.

Tabla 3.5 Especificación de recursos y condiciones del entorno de pruebas

JMeter	RNF del SIGM	Condiciones PC
Máquina Virtual Java 1.3	Software Servidor de aplicación: - Sistema Operativo Linux, y servidores Apache 2.0 o una versión superior y MySQL 5.0 una versión superior. - Internet Explorer 5.5 o superior. PC cliente: - Se deberá disponer del Sistema Operativo Linux o Windows 98 o una versión superior. Hardware: - Capacidad mínima con	Se emplearon para la ejecución de las pruebas 2 PC con las siguientes características. 1.- Sistema Microsoft Windows XP Professional Versión 2002 Service Pack 3 2.- Intel (R) Pentium 5 KPL-VM Dual CPU 2.00 GHz, 0.99 GB de RAM

	128 MB de RAM Servidor de aplicación: - Pentium a 333 MHz o superior. - 512 MB RAM o superior. -10 GB de espacio libre en Disco Duro. Para el client page: - Pentium a 233 MHz o superior. - 64 MB RAM o superior. - Tarjeta de red (MODEM o red con TCP-IP para conexión al servidor).	

En la tabla anterior se plasman las condiciones del entorno de pruebas en JMeter, así como los requerimientos no funcionales de hardware y software del sistema, declarados en la plantilla de Especificación de requisitos. Se considera importante especificar bajo qué circunstancias la aplicación se ejecuta, incluyendo la cantidad de personal asociada a la misma. Si las pruebas serán ejecutadas en varias PC, especificar el servidor de la aplicación y las características de la misma para que ésta se pueda ejecutar a modo servidor. Para un eficiente análisis de los resultados se deben tener en cuenta el entorno en que se ejecutaron las pruebas y el ambiente real donde se desplegará el sistema; si ambas condiciones son lo más cercanas posibles, los resultados serán satisfactorios y consecuentes.

3.3.1 Resumen de los resultados de la ejecución de las pruebas

Como se especificó en el capítulo anterior en la descripción del proceso de automatización de las pruebas, se realizaron 6 iteraciones que simulaban el acceso concurrente de 100, 200 y 300 usuarios respectivamente a la aplicación, con el servidor sobre el sistema operativo Linux y Windows. Se mezclaron las grabaciones de todos los escenarios, con la intención de obtener resultados generales del sistema; de manera que se simulara el acceso de las cantidades mencionadas distribuido en los diferentes escenarios de los casos de uso. De este modo se comprobó el comportamiento del sistema

para un número de usuarios inferior al estimado, la estimación propiamente definida (200 usuarios) y una cantidad superior, representando una sobrecarga de conexiones simultaneas según el estimado proporcionado por el equipo de desarrollo, con el objetivo de estresar el sistema.

Servidor de aplicación sobre Sistema Operativo Windows

Analizando la iteración para 100 usuarios conectados en un ciclo y un período de subida de 1 segundo, se obtuvieron los siguientes resultados del Informe Agregado:

Tabla 3.6 Resumen del Informe Agregado obtenido para 100 usuarios

Elemento	# Muestras	Media	Mediana	Línea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
SIGM	1983	10544	922	30688	0	57250	4.89	5.2/sec	44732.2

De la misma forma, pero para un total de 200 usuarios, se obtuvieron los siguientes datos:

Tabla 3.7 Resumen del Informe Agregado obtenido para 200 usuarios

Elemento	# Muestras	Media	Mediana	Línea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
SIGM	446	27763	21000	92656	0	128141	10.09	1.5/sec	16704.8

Luego de ejecutar las pruebas para simular las peticiones realizadas por 300 usuarios concurrentes, a todos los escenarios propuestos, se recogieron los resultados que a continuación se muestran:

Tabla 3.8 Resumen del Informe Agregado obtenido para 300 usuarios

Elemento	# Muestras	Media	Mediana	Línea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec

SIGM	550	28990	9328	108157	0	110094	8.73	1.6/sec	17222.6

Una vez ejecutadas las pruebas a los diferentes escenarios, se puede resumir que para un estimado de 200 usuarios conectados simultáneamente (siendo estos lo que debe soportar el sistema), se registró un total de 446 peticiones con un tiempo de respuesta de 21 000 milisegundos; mientras que para 100 y 300 usuarios concurrentes, se arrojó en ambos casos un número mayor de peticiones respecto al correspondiente a los 200 usuarios, con un tiempo de respuesta y un % de error de las páginas menor. De igual manera, el rendimiento y la velocidad de carga son mayores en los casos límites aun teniendo un número superior de peticiones. Como se observa, estos datos son contradictorios, no se concibe que con un número mayor de peticiones se obtenga un tiempo de respuesta inferior al registrado con los 200 usuarios estimados para un buen funcionamiento del sistema, lo mismo ocurre con los demás parámetros medidos. De manera general se concluye que a medida que aumenta el número de usuarios disminuye el rendimiento de la aplicación.

Servidor de aplicación sobre Sistema Operativo Linux

Considerando un estimado de 100 usuarios, con un período se subida de 0 segundos, se recogieron los resultados que se muestran a continuación:

Tabla 3.9 Resumen del Informe Agregado obtenido para 100 usuarios

Elemento	# Muestras	Media	Mediana	Línea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
SIGM	207	12692	13875	23703	328	37672	0.0	1.6/sec	8716.8

De la misma forma, pero para un total de 200 usuarios, se obtuvieron los siguientes datos:

Tabla 3.10 Resumen del Informe Agregado obtenido para 200 usuarios

Elemento	# Muestras	Media	Mediana	Línea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
SIGM	265	20498	15844	54000	343	61375	1.89	1.4/sec	6409.3

Luego de ejecutar las pruebas para simular las peticiones realizadas por 300 usuarios concurrentes, a todos los escenarios propuestos, se recogieron los resultados que a continuación se muestran:

Tabla 3.11 Resumen del Informe Agregado obtenido para 300 usuarios

Elemento	# Muestras	Media	Mediana	Línea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
SIGM	417	23551	18312	45625	343	70984	0.0	1.0/sec	5122.0

Como se observa en los índices previamente expuestos luego de la ejecución de las pruebas, esta vez con el servidor de aplicación montado en Linux, los resultados obtenidos son satisfactorios. A medida que se aumentó la cantidad de usuarios concurrentes se elevó el número de peticiones, lo que se corresponde con un incremento en el tiempo de respuesta. El % de error es notablemente menor, lo que indica que la mayoría de las páginas se cargaron satisfactoriamente. Se puede concluir que el rendimiento del software sobre este sistema operativo es mejor que sobre Windows, aunque queda evidenciado que para una sobrecarga de usuarios concurrentes el sistema disminuye su rendimiento, y el desempeño en términos de velocidad de carga de las páginas merma notablemente. Por tanto se recomienda que no se deban conectar simultáneamente al sistema más de 200 usuarios, y en caso de ser así, las condiciones de despliegue del servidor de aplicación deberían optimizarse.

Advertencia

Durante las pruebas, se registró un % de error mayor en Windows que en Linux, plasmado en las tablas resúmenes correspondientes. Dicho error se reconoció como: /js/tiny_mce/plugins/flash/editor_plugin.js, que representa una carpeta donde se guardan estilos para tablas en código java script, y básicamente simboliza una página que no ha sido carga da satisfactoriamente. En las siguientes figuras aparece reflejado la reiterada aparición de este fallo para ambos sistemas operativos en el Árbol de Resultados que genera la herramienta JMeter.

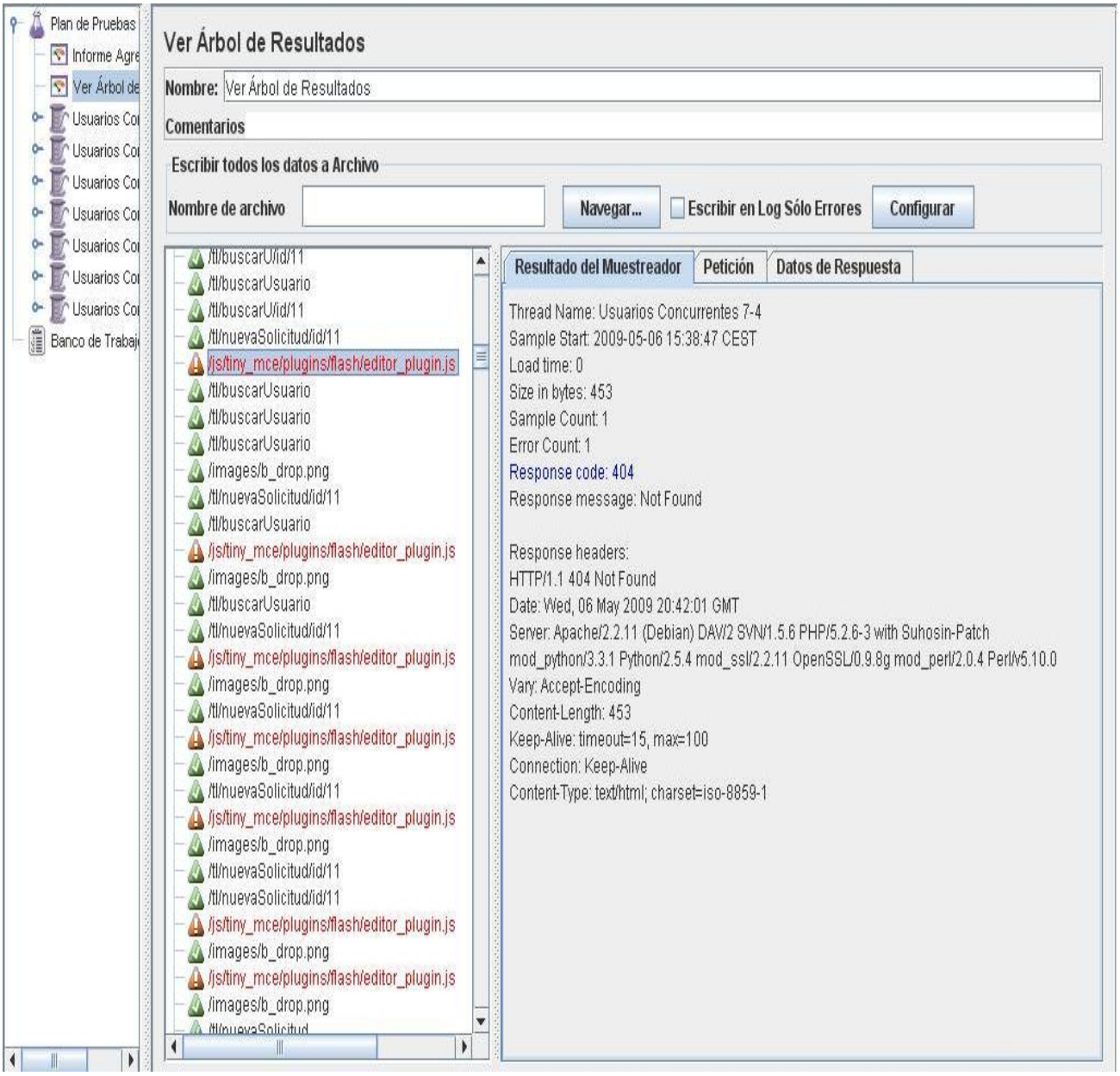
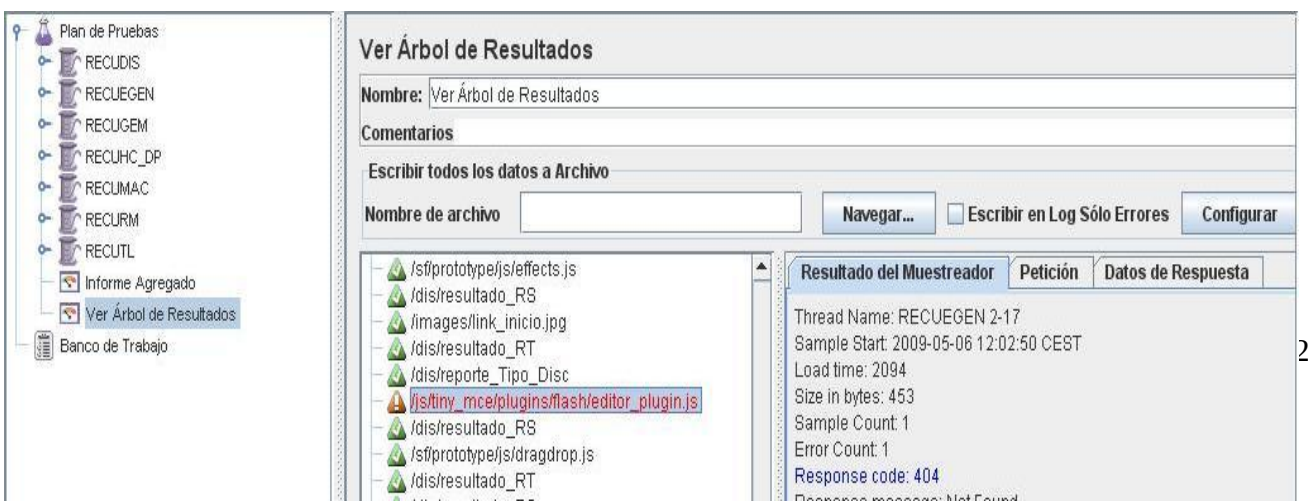


Ilustración 3.1 Error en Windows



3.5 Evaluación del producto SIGM

Luego de efectuar el proceso de pruebas de liberación al software SIGM, se considera objetivo realizar una evaluación del producto atendiendo a los atributos de calidad:

- ✓ Eficiencia
- ✓ Seguridad
- ✓ Usabilidad
- ✓ Confiabilidad
- ✓ Portabilidad

Una síntesis de la evaluación se representa en el siguiente gráfico:

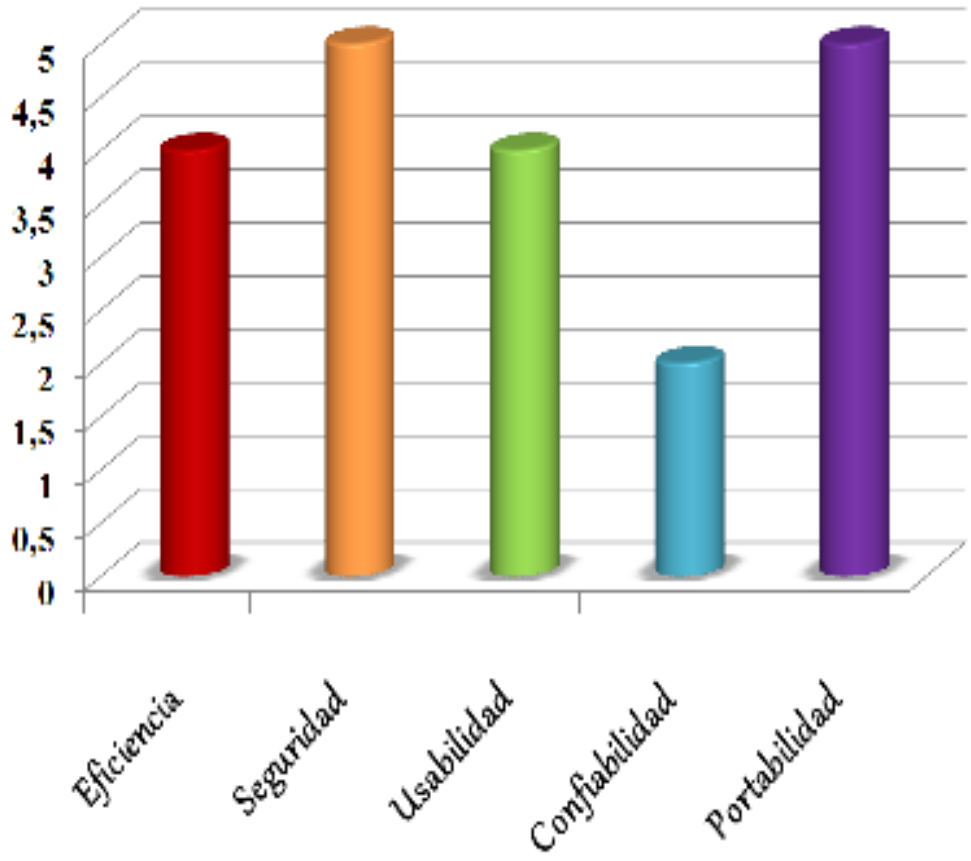


Ilustración 3.3 Evaluación del SIGM mediante atributos de calidad

La valoración consumada se sustentó en una lista de chequeo (ver expediente de proyecto). Esta lista contiene una serie de preguntas cuyas respuestas consisten en una evaluación que se le confiere a cada uno de estos parámetros de calidad por separado. Dicha evaluación se concede en dependencia del grado de disponibilidad de las propiedades medidas, según las preguntas. Esto es, un parámetro evaluado de 0 puntos, se le atribuye a una propiedad no disponible, lo que puede significar una función o característica ausente en el sistema. Una puntuación de 2 representa una disponibilidad parcial de la propiedad, es decir, en parte son satisfactorios los resultados observados pero sigue habiendo fallos en algunos casos. En cambio una calificación de 4 describe un parámetro correctamente implementado. El 5 es para la excelencia, aquel parámetro que no tenga ningún señalamiento negativo y no presente ningún fallo.

En el caso de la Usabilidad se tuvo en cuenta un total de 17 preguntas, de las cuales siete resultaron evaluadas de 5 puntos, seis de 4, dos de 2 y dos de 0; por cuanto analizando cualitativamente estos índices se determinó que para definir una calificación final excelente (5 puntos) no se admiten aspectos evaluados con 0 ni con 2. Dicho esto, se evalúa la característica de Usabilidad del software de 4 puntos, coincidiendo este resultado con el del análisis cuantitativo en el que se promedian la suma de todos los puntos por aspectos.

En los casos particulares de Seguridad y Portabilidad, se evalúan de 5 puntos, ejerciendo el mismo análisis anterior.

Para evaluar la Eficiencia se tuvo presente el tiempo de respuesta requerido para la aplicación, siendo éste evaluado de 4 puntos.

Finalmente, se considera que el producto no posee un grado de Confiabilidad óptimo, debido a que el sistema no se recupera de manera total ante los fallos ocurridos, por lo que se determina una evaluación de 2 puntos para este atributo.

Conclusiones del capítulo

Una vez desarrollado el proceso de pruebas de liberación al SIGM, se procede a realizar un análisis cuantitativo y cualitativo de los parámetros de evaluación medidos para valorar el software y verificar la calidad del mismo. Las pruebas de liberación, dígase aplicación y revisión de la documentación arrojaron un total de 160 y 157 no conformidades respetivamente, registrándose un seguimiento de las mismas en su correspondiente plantilla y formando parte de los casos de prueba elaborados durante el proceso. En el caso de las pruebas de Carga y Estrés, si bien no se ejecutaron durante la liberación del producto, se obtuvieron índices estadísticos que reflejaron déficit en el rendimiento y desempeño del software para condiciones anormales y un aumento gradual de la carga de los usuarios que podía soportar. Estos resultados son de gran importancia a tener en cuenta por el equipo de desarrollo para las próximas versiones del sistema, para darle seguimiento a los fallos encontrados y las posibles causas de los mismos, lo que facilitaría el tratamiento de errores y la prevención de defectos. Para evaluar el producto, se utilizó la lista de chequeo de atributos de calidad, razón por la cual se le proporciona una evaluación al software en atendiendo a estos parámetros de manera individual, no así de manera general. Analizado esto se concluye que el SIGM es un software *portable*, *seguro* en términos de acceso a la información precisa muy bien delimitado, *usable* atendiendo a las características de su interfaz, *eficiente*, pero parcialmente confiable, dado los fallos que presentó y su insuficiente recuperación con respecto a estos.

CONCLUSIONES

Para desarrollar el proceso de liberación del SIGM:

- ✓ Se diseñaron Casos de Prueba basados en los casos de uso del sistema.
- ✓ Se aplicaron diferentes técnicas de pruebas, especialmente las pruebas funcionales a la aplicación.
- ✓ Se empleó el método de Caja Negra mediante la técnica de Partición de Equivalencia.
- ✓ Los resultados obtenidos de las pruebas a la aplicación y la documentación, se registraron en la plantilla de No Conformidades.
- ✓ Para medir el rendimiento del sistema se ejecutaron pruebas de Carga y Estrés con la herramienta JMeter.
- ✓ Se evalúo el software atendiendo a los atributos de calidad.

RECOMENDACIONES

Posterior a la realización de todas pruebas concebidas para liberar el SIGM, y dados los resultados obtenidos durante este proceso, se recomienda que:

- ✓ Se fomente en los proyectos productivos la realización de las pruebas en todo el ciclo de vida del software, de manera tal que se detecte el mayor número de errores previamente al proceso de pruebas de liberación.
- ✓ En el grupo de desarrollo los aseguradores de la calidad garanticen el diseño y la ejecución de todos los casos de pruebas, a modo de avalar un proceso de pruebas eficiente.
- ✓ Para los grupos de calidad de todas las facultades se incluya dentro del proceso de pruebas de liberación, las pruebas de Carga y Estrés automatizadas con la herramienta JMeter, dado que con las pruebas funcionales no se asegura un rendimiento y desempeño del sistema una vez desplegado en la entidad que lo solicita.
- ✓ En el equipo de desarrollo de los proyectos se detallen con datos específicos los requerimientos de rendimientos del sistema implementado, para de esta manera optimizar los resultados obtenidos con las pruebas de Carga y Estrés.

REFERENCIAS BIBLIOGRÁFICAS

1. Hugo Vázquez, Roberto. Taller de Calidad de Software: Introducción a la Calidad de Software. [En línea: 26/03/2006] [Citado el: 26/11/2008]
   Disponible en:
   http://gridtics.frm.utn.edu.ar/docs/Introduccion%20a%20la%20Calidad%20de%20Software%20Vazquez.pdf.

2. Hugo Vázquez, Roberto. Taller de Calidad de Software: Introducción a la Calidad de Software. [En línea: 26/03/2006] [Citado el: 26/11/2008]
   Disponible en:
   http://gridtics.frm.utn.edu.ar/docs/Introduccion%20a%20la%20Calidad%20de%20Software%20Vazquez.pdf

3. Pressman, Roger S. Ingeniería del Software. Un enfoque práctico. 5ta ed. 2002. [Citado el: 05/12/2008]
   También disponible en:
   http://bibliodoc.uci.cu/pdf/reg02689.pdf.

4. Conferencia # 5 Pruebas. Ingeniería del Software II. [Citado el: 05/12/2008]
   Disponible en:
   http://teleformacion.uci.cu

5. Juristo Natalia, Moreno Ana M., Vegas Sira. Técnicas de evaluación de software. [Citado el: 06/12/2008] [En línea: 17/10/2006].
   Disponible en:
   http://teleformacion.uci.cu.

6. Blank Isabel, Herrera Larissa, Ortiz Miguel. Pruebas de funcionalidad. [Citado el: 6/12/2008] [En línea: 05/2005].
   Disponible en:
   http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf.

7. Maurer, Donna. ¿Qué es usabilidad y cómo conseguirla? [Citado el: 07/12/2008] [En línea: 17/10/2007].
   Disponible en:

http://www.sg.com.mx/content/view/405

8. Zibert van Gricken, Carolina. Pruebas de Confiabilidad. [En línea: 30/05/2005] [Citado el: 07/12/2008]
   Disponible en:
   http://carolina.terna.net/ingsw3/datos/Pruebas_de_Confiabilidad.pdf

9. Atencia Yépez, Amaya. Análisis de fiabilidad de sistemas aplicando técnicas de crecimiento de fiabilidad del software. [En línea: 09/2007] [Citado el: 07/12/2008].
   Disponible en:
   www.aemes.org/rpm/descarga r.php?volumen=4&numero=3&articulo=1

10. Hill Deris, Monteverde Alejandro. Pruebas de soportabilidad. [En línea: 01/06/2005] [Citado el: 08/12/2008]
    Disponible en:
    http://carolina.terna.net/ingsw3/datos/Pruebas_de_Soporte.pdf

11. Ingeniería de Software "Pruebas". [Citado el: 08/12/2008]
    Disponible en:
    http://delfin.mxl.uabc.mx/~angelica/Pruebas.pdf

12. Sánchez Almenares, Luidmila. Prueba Automática de Carga y Estrés en el Proyecto CICPC. Trabajo de Diploma para optar por el título en Ciencias Informáticas. Universidad de las Ciencias Informáticas, Ciudad de la Habana, 2008. [Citado el: 10/01/2009]

13. Mejoramiento del Proceso de Pruebas y Corrección de Defectos de Software en un ambiente globalizado. [En línea 2009] [Citado el: 11/01/2009]
    Disponible en:
    http://chie.uniandes.edu.co/~gsd/index.php?option=com_content&task=view&id=129&Itemid=183

14. Software Quality Systems S.A, Validación y Verificación. [En línea 2009] [Citado el: 13/04/09].
    Disponible en:
    http://www.sqs.es/es/services/validation.php

15. Javier Acuña, César. Pruebas de Software. Ingeniería de Software I.
    Disponible en:

http://kybele.escet.urjc.es/Documentos/ISI/Pruebas%20de%20Software.pdf

16. Mendoza Sánchez, María A. Metodologías de Desarrollo de Software. [En línea:07/06/2004] [Citado: 10/01/2009].
    Disponible en:
    http://www.informatizate.net

17. Parra Infante, Yanet y Rodríguez Ramírez, Ricardo. Sistema Automatizado Cubano para Control de Equipos Médicos: Módulo de gestión de información de equipos médicos de Radiofísica. Trabajo de Diploma para optar por el título en Ciencias Informáticas. Universidad de las Ciencias Informáticas, Ciudad de la Habana, 2008.

18. Free Download Manager. TestV (vTest) [En línea 06/04/2007] [Citado: 22/03/2009].
    Disponible en:
    http://www.freedownloadmanager.org/es/downloads/pruebas_funcionales_gratis/

19. Free Download Manager. AdventNet QEngine [En línea 22/08/2006] [Citado: 22/03/2009].
    Disponible en:
    http://www.freedownloadmanager.org/es/downloads/pruebas_funcionales_gratis/

20. Free Download Manager. AdventNet QEngine WebTest [En línea 18/02/2006] [Citado: 22/03/2009].
    Disponible en:
    http://www.freedownloadmanager.org/es/downloads/pruebas_funcionales_gratis/

21. Próxima parada. El Conocimiento. Selenium IDE, una herramienta para realizar pruebas de aplicaciones web. [En línea 24/04/2008] [Citado: 22/03/2009].
    Disponible en:
    http://dacosta51.wordpress.com/2008/04/24/selenium-ide-una-herramienta-para-realizar-pruebas-de-aplicaciones-web/

BIBLIOGRAFÍA

1. Nieves Borrero Martha, Góngora Rodríguez Asnier. Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad de Software: Módulo Gestión de las No Conformidades. Trabajo de Diploma para optar por el título en Ciencias Informáticas. Ciudad de la Habana, 2007.
2. Saavedra López Lesdey, Pupo Blez Yohandris. Diseño y aplicación de pruebas al producto Registro Cubano de Discapacitados. Trabajo de Diploma para optar por el título en Ciencias Informáticas. Ciudad de la Habana, 2007.
3. Claro Arceo, Alfonso. Sistema Informático de Genética Médica. Informe del XVI de Ciencia y Técnica. Universidad de las Ciencias Informáticas, Ciudad de la Habana, 2008.
4. Hugo Vázquez, Roberto. Taller de Calidad de Software: Introducción a la Calidad de Software. [En línea: 26/03/2006]
   Disponible en:
   http://gridtics.frm.utn.edu.ar/docs/Introduccion%20a%20la%20Calidad%20de%20Software%20Vazquez.pdf.
5. Hugo Vázquez, Roberto. Taller de Calidad de Software: Introducción a la Calidad de Software. [En línea: 26/03/2006]
   Disponible en:
   http://gridtics.frm.utn.edu.ar/docs/Introduccion%20a%20la%20Calidad%20de%20Software%20Vazquez.pdf
6. Pressman, Roger S. Ingeniería del Software. Un enfoque práctico. 5ta ed. 2002
   También disponible en:
   http://bibliodoc.uci.cu/pdf/reg02689.pdf.
7. Juristo Natalia, Moreno Ana M., Vegas Sira. Técnicas de evaluación de software. [En línea: 17/10/2006].
   Disponible en:
   http://teleformacion.uci.cu.
8. Blank Isabel, Herrera Larissa, Ortiz Miguel. Pruebas de funcionalidad. Mayo, 2005
   Disponible en:
   http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf.
9. Maurer, Donna. ¿Qué es usabilidad y cómo conseguirla? [En línea: 17/10/2007].
   Disponible en:
   http://www.sg.com.mx/content/view/405
10. Zibert van Gricken, Carolina. Pruebas de Confiabilidad. [En línea: 30/05/2005]

Disponible en:
http://carolina.terna.net/ingsw3/datos/Pruebas_de_Confiabilidad.pdf

11. Atencia Yépez, Amaya. Análisis de fiabilidad de sistemas aplicando técnicas de crecimiento de fiabilidad del software. [En línea: 09/2007]
    Disponible en:
    www.aemes.org/rpm/descarga r.php?volumen=4&numero=3&articulo=1
12. Hill Deris, Monteverde Alejandro. Pruebas de Soportabilidad. [En línea: 01/06/2005]
    Disponible en:
    http://carolina.terna.net/ingsw3/datos/Pruebas_de_Soporte.pdf
13. Ingeniería de Software “Pruebas”.
    Disponible en:
    http://delfin.mxl.uabc.mx/~angelica/Pruebas.pdf
14. Sánchez Almenares, Luidmila. Prueba Automática de Carga y Estrés en el Proyecto CICPC. Trabajo de Diploma para optar por el título en Ciencias Informáticas. Universidad de las Ciencias Informáticas, Ciudad de la Habana, 2008.
15. Mejoramiento del Proceso de Pruebas y Corrección de Defectos de Software en un ambiente globalizado.
    Disponible en:
    http://chie.uniandes.edu.co/~gsd/index.php?option=com_content&task=view&id=129&Itemid=183
16. Sánchez Almenares, Luidmila. Manual Cómo realizar pruebas de Carga y Estrés. Universidad de las Ciencias Informáticas, 2008.
17. Javier Acuña, César. Pruebas de Software. Ingeniería de Software I.
    Disponible en:
    http://kybele.escet.urjc.es/Documentos/ISI/Pruebas%20de%20Software.pdf
18. Huerta, Rosendo. Proceso de Análisis Integral de Disponibilidad y Confiabilidad como Soporte para el Mejoramiento Continuo de las Empresas. Monterey, México, 2006.
    Disponible en:
    http://www.noria.com/sp/rwla/conferencias/mem/Paper%20Rosendo.pdf
19. Sarno Severi, Emilio Giuseppe. Como puede obtener la Confiabilidad que usted necesita de sus activos. Cartagena de Indias, Colombia, 2007.
    Disponible en:
    http://www.andeanenergycongress.com/presentaciones/dia_02/ESS.pdf
20. Espinoza Bárbara, Quinta Vanessa, Vegas Alexandra. Pruebas de rendimiento. Julio, 2005

Disponible en:
http://carolina.terna.net/ingsw3/datos/Pruebas_de_Desempe%F1o.pdf

21. Soriano, Amelia. Tipos de Prueba. Mayo, 2005.
    Disponible en:
    http://carolina.terna.net/ingsw3/datos/Tipos_Prueba.pdf
22. Visual Studio Team System 2008 Test Developer Center. Trabajar con Pruebas de Carga.
    Disponible en:
    http://msdn.microsoft.com/es-es/library/ms182561(VS.80).aspx
23. Osmosis Latina. Guia JMeter. [En línea: 10/05]
    Disponible en:
    http://www.osmosislatina.com/jmeter/
24. Vicente Toribio, José María. Tutorial JMeter. Abril, 2005
    Disponible en:
    http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jmeter