Facultad 5 “Centro de Desarrollo de Informática Industrial”

Trabajo de diploma para optar por el Título de
Ingeniero en Ciencias Informáticas.

Título:

Automatización de pruebas de carga y estrés del Sistema de Información de Perforación de Pozos Petroleros (SIPP).

Autores:
Dayana Víctores Carmona
Yanisleydis Pelier Escalona

Tutores:
Ing. Amado Espinosa Hidalgo
Ing. Mariela Cepero Núñez

Cotutor:
Ing. Junior Muñoz Treto.

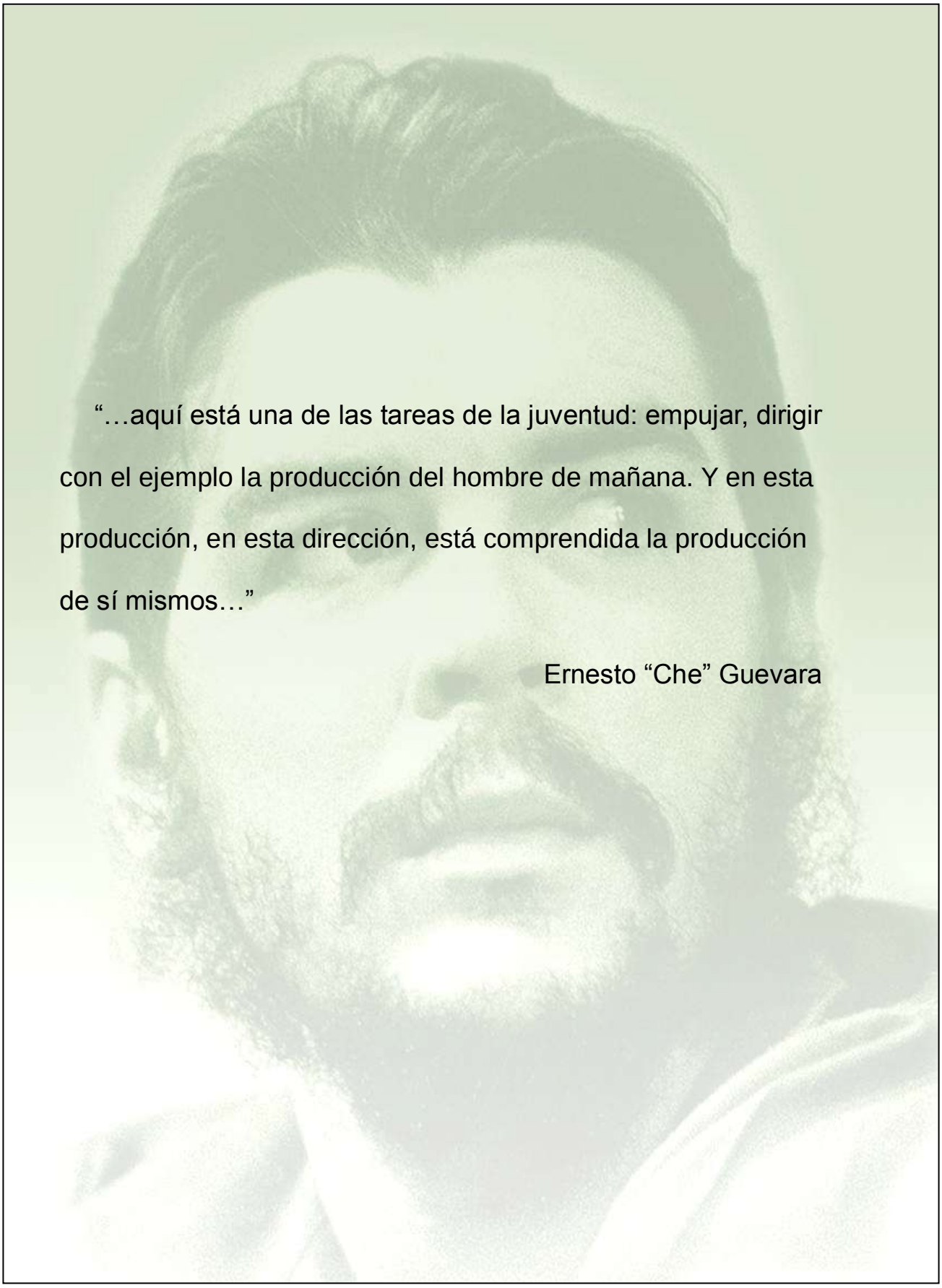
Ciudad de La Habana, Junio de 2010

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_____.

Dayana Víctores Carmona	Yanisleydis Pelier Escalona
_____	_____
Firma del Autor	Firma del Autor
Amado Espinosa Hidalgo	Mariela Cepero Núñez
_____	_____
Firma del Tutor	Firma del Tutor



"…aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos…"

Ernesto "Che" Guevara

DATOS DE CONTACTO

Amado Espinosa Hidalgo.

Ingeniero Informático y profesor asistente del Departamento de Ingeniería y Gestión de Software de la Facultad 5. Posee 7 años de experiencia en la actividad docente y productiva.

Correo electrónico: aespinosa@uci.cu

Mariela Cepero Núñez

Ingeniera en Ciencias Informáticas perteneciente al Departamento Integración y Despliegue del Centro de Informática Industrial. Profesor instructor con 3 años de experiencia docente y 4 años de experiencia en la producción.

Correo electrónico: mceperon@uci.cu

Agradecimientos

Agradecimientos de Dayana:

Quiero agradecerle ante todo a mi mamá por darme la vida, sin ella nada hubiera sido posible.

Agradezco a mis dos padres; a Tony (mi papá) por contribuir en mi formación y a Rolandito por estar a mi lado desde que tengo nueve meses apoyándome en todo.

A mi familia en general, en especial a mi tío Noel porque sin su colaboración todo sería un poco más difícil, a mi tía Yamila por estar conmigo en las buenas y las malas, a mi tita Sonia y Mayda por su preocupación y ayuda constante y a mis hermanos Roly y Yoan y primos Haysel y Javier, por ver en mi un ejemplo.

Mi gratitud a Trini y Abreu por su apoyo incondicional.

A amigos como Alién, Riolvis, Dayron, Ernesto, Yunetsy, Ailem y Yanisleydis por ayudarme en los momentos más difíciles a lo largo de estos cinco años sin reclamar nada a cambio.

No puedo dejar de agradecerle a Arlenys, Juan Gualberto, Liuber, Leandro y Yuniel que no han dejado de ayudarme cuando los he necesitado.

A mis tutores Amado y Mariela mil gracias, así como a la profesora Yadira Ramírez.

A todas las personas que contribuyeron en mi educación, mi gratitud a Fidel y la Revolución que me permitieron estudiar y convertirme en alguien útil para la sociedad.

Dedicatoria

Agradecimientos de Yanisleydis

Agradezco:

A mis padres Lyla y Rafael por el apoyo incondicional durante toda mi vida, por su amor, por ser mi fuente de inspiración diaria y por confiar en mí. A mi madre por enseñarme a ser mejor persona cada día y a Rafael, por asumir el papel de ser mi padre desde pequeña. A ambos por jugar un papel tan importante en mi vida.

A mi abuelita Esperanza, a quien quiero mucho, gracias por apoyarme siempre y guiarme por el buen camino.

A mi novio que siempre ha estado a mi lado, gracias por su apoyo, por ayudarme, quererme, brindarme estabilidad y ayudarme a superar las dificultades por las que he pasado en estos 5 años. Gracias nene.

A mi familia en general, en especial a mi tía Elsita que me ha acogido en su casa como una hija más y a mi prima Marian por siempre tener una sonrisa para mí.

A mi amiga y compañera de tesis Dayana por su ayuda, apoyo en todos los días de trabajo, desvelos y preocupaciones vividas durante la realización de la tesis.

A nuestros tutores Amado Espinosa y Mariela Cepero y al cotutor Junior Muñoz por su apoyo durante la investigación y las críticas constructivas para una mejor realización del presente trabajo.

A Yunetsy por toda su ayuda y a Alien por su apoyo incondicional, cuando más lo necesitábamos.

A la Universidad y su colectivo de profesores que me han formado durante estos cinco años .En especial a la profesora Yadira y al profesor Manuel Villanuevas.

A mis amigos y amigas de Moa, que en todo este tiempo me han acompañado y me han brindado su apoyo siempre, en especial a Darilkis , por estar siempre en las buenas y malas.

A Claudia y Osvaldito por la atención que me han prestado desde que los conocí. Les estaré eternamente agradecida.

A Yaylín, Adriana y Orlis por brindarme su comprensión y amistad.

A los muchachos y muchachas del 5107 con los que pasé casi toda mi carrera y a los actuales del 5507.

A todas las amistades que he conocido en la escuela.

A la Revolución y a Fidel por darme la oportunidad de realizar este sueño.

En fin, quiero agradecer a todas aquellas personas que de una forma u otra han contribuido al desarrollo de este trabajo de diploma y a mi formación tanto profesional como personal.

Dedicatoria

Dedicatoria de Dayana:

Dedico el trabajo de diploma a mis abuelos Eva de la Nuez Morales y Antonio Victores Gómez.

Dedicatoria de Yanisleydis:

Dedico todo mi esfuerzo durante estos cinco años a mis padres Lyla y Rafael y a mi abuelita Esperanza, por ser quienes me han enseñado todo en la vida, los que siempre me han apoyado y han confiado en mí. Los quiero mucho.

Resumen

Resumen

La investigación se centra principalmente en describir un procedimiento para la realización de pruebas de carga y estrés a través de la elección de las herramientas OpenLoad y Hammerhead mediante aspectos comparativos relevantes. El proceso descrito en el documento está enmarcado en demostrar que las pruebas automatizadas aplicadas al Sistema de Información de Perforación de Pozos Petroleros (SIPP) constituyen la solución más eficaz para demostrar que la aplicación cumple con los criterios de rendimiento que determinan el comportamiento de la misma ante una carga determinada y además permite someterla al límite de su funcionamiento para determinar su robustez y evaluar la capacidad de resistencia del sistema ante posibles vulnerabilidades.

Palabras Claves

Pruebas de carga, pruebas de estrés, pruebas automatizadas, rendimiento.

Índice

Índice

Resumen VIII

Introducción. 1

Capítulo 1: Fundamentación Teórica. 4

1.1 Introducción.....4

1.2 Pruebas de software.....4

1.2.1 Etapas involucradas en las pruebas de software. 5

1.2.2 Estrategia de prueba.....6

1.2.3 Ventajas de probar el software.....8

1.3 Metodologías de Desarrollo de Software. 9

1.3.1 Metodologías tradicionales. 9

1.3.2 Metodologías ágiles. 9

1.3.3 Proceso Unificado del Desarrollo de Software. 10

1.4 Automatización de pruebas. 17

1.4.1 Herramientas para pruebas automatizadas.....19

1.5 Proyecto SIPP.....27

Conclusiones Parciales.....28

Capítulo 2: Proceso automatizado para pruebas de carga y estrés empleando OpenLoad y Hammerhead.....29

2.1 Introducción.....29

2.2. Estrategia para la realización de pruebas de carga y estrés.....29

2.2.1. Alcance.....29

2.2.2 Roles identificados.....30

Índice

2.2.3 Objetivos. 30

2.2.4. Fases a desarrollar. 31

Conclusiones Parciales. 48

CAPITULO 3: VALIDACIÓN DEL PROCEDIMIENTO. 49

3.1 Introducción. 49

3.2 Tipos de evaluación. 49

3.3 Selección del método. 49

3.4 Encuesta. 51

3.5 Selección del grupo de expertos. 52

3.6 Coeficiente de Concordancia de Kendall y prueba de hipótesis. 52

3.6.1 Análisis de la concordancia. 53

3.6.2 Análisis de los resultados. 55

Conclusiones Parciales. 56

Conclusiones Generales. 57

Recomendaciones 58

Referencias Bibliográficas. 59

Bibliografía. 62

Anexos. 64

Glosario de Términos. 67

Índice de figuras

Figura 1 Fases, Flujo de trabajo de RUP. 11

Figura 2 Marco de pruebas automatizadas. 19

Figura 3 Estructura del Procedimiento. 31

Figura 4 Actividades por roles Fase de planificación. 32

Figura 5 Actividades por roles Fase de Diseño. 35

Figura 6 Actividades por roles Fase Implementación de componentes. 38

Figura 7 Actividades por roles Fase Ejecución. 39

Figura 8 Actividades por roles Fase Cierre. 42

Figura 9 CP1_Mostrar uso de la barrena_Carga. 43

Figura 10 CP2_Mostrar cronograma de perforación_Carga. 43

Figura 11 CP3_ Reporte diario de perforación_Carga. 44

Figura 12 CP4_Costo diario (Listar Modificar) _Carga. 44

Figura 13 Resumen de Pruebas de Carga 45

Figura 14 CP1_Mostrar uso de la barrena_Estrés. 45

Figura 15 CP2_Mostrar cronograma de perforación_ Estrés. 46

Figura 16 CP3_ Reporte diario de perforación_ Estrés. 46

Figura 17 CP4_Costo diario (Listar Modificar) _ Estrés. 47

Figura 18 Resumen de las Pruebas de Estrés. 47

Figura 19 Proceso Delphi. 50

Figura 20 Fórmulas para el cálculo del coeficiente de Kendall. 52

Figura 21 Promedio por criterios de evaluación. 55

Figura 22 Grado de unicidad de las evaluaciones por criterios. 55

Índice de tablas

Tabla 1 Tipos de pruebas propuestas por RUP. 17

Tabla 2 Comparación de las herramientas de carga. 25

Tabla 3 Comparación de las herramientas de estrés. 27

Tabla 4 Roles para el equipo de pruebas. 30

Tabla 5 Entorno de pruebas. 34

Tabla 6 Resultados de las evaluaciones de los expertos. 54

Tabla 7 Rangos de puntaje ligados. 54

Tabla 8 Cálculo de S. 54

Tabla 9 Cálculo de T. 54

Introducción

Introducción.

Las pruebas de software en aplicaciones han evolucionado, no sólo permiten depurar la aplicación, sino que han pasado a convertirse en un proceso integrador, automatizable y administrable que se realiza desde el momento del inicio de la codificación y se extiende hasta la entrega final al cliente.

En nuestro país se comienza a profundizar en la realización de pruebas al software para que su funcionamiento sea el esperado por el cliente durante su explotación. La Universidad de Ciencias Informáticas es una de las proveedoras de software en el país, la misma cuenta con una Infraestructura formada por Centros Productivos que desarrollan proyectos temáticos ubicados en 9 Facultades.

En la Facultad 5 se encuentra el Centro de Desarrollo de Informática Industrial (CEDIN), en el cual se ha desarrollado un Sistema de Información de Perforación de Pozos Petroleros (SIPP) con el objetivo de controlar y asegurar el flujo de información que se genera en los pozos en perforación. Es una aplicación distribuida, que tiene una alta demanda, por lo que está en constante desarrollo y evolución. La misma permite tener una parte en el Pozo y la otra en la Dirección de Intervención y Perforación de Pozos (DIPP) haciendo el sistema más ligero y seguro.

Por tanto, a partir de todo lo señalado se puede precisar la **situación problémica** de la siguiente manera: a pesar de que en el centro existe un equipo encargado de realizar las pruebas a los productos desarrollados, el proceso clave de pruebas de carga y estrés en el SIPP se realiza de forma insuficiente, pues este equipo desconoce cómo emplear las herramientas necesarias y efectivas para ese control, lo cual trae como consecuencia la incorrecta verificación de los requerimientos de rendimiento de la solución, bajo condiciones de tensión en distintos escenarios, la incapacidad de predecir el comportamiento de la aplicación en cuanto a escalabilidad, fiabilidad , uso de los recursos y su solidez en los momentos de carga extrema, una vez desplegadas en las instalaciones de los clientes.

Introducción

Analizando la situación problémica expuesta con anterioridad se define como ***problema de investigación:*** ¿Cómo el insuficiente desarrollo de las pruebas de carga y estrés en el SIPP inciden en la verificación incorrecta de los requerimientos de rendimiento de la solución y en la incapacidad de predecir el comportamiento de la aplicación?

Teniendo en cuenta el problema de investigación que precisa como ***objeto de estudi*o:** Las pruebas de software de rendimiento.

Por lo cual se deriva como ***campo de acción***: Las pruebas de carga y estrés en aplicaciones web.

Por todo lo anterior se plantea como ***Objetivo general*** el siguiente**:**

Desarrollar un procedimiento para emplear herramientas automatizadas durante las pruebas de carga y estrés al SIPP.

Para lograr el objetivo de esta investigación y dar solución a la problemática antes expuesta, es necesaria la realización de las siguientes ***tareas de investigación*:**

- ✓ Estudio del estado del arte centrado en pruebas de software y herramientas para la automatización de pruebas de carga y estrés en aplicaciones web.
- ✓ Selección de las herramientas candidatas de acuerdo a sus principales funcionalidades y características.
- ✓ Selección de los procedimientos existentes a nivel nacional e internacional para desarrollar pruebas automatizadas a aplicaciones web.
- ✓ Elaboración del informe que fundamenta la propuesta.
- ✓ Realización de ejemplos de aplicación en calidad de prueba de concepto.
- ✓ Diseño de los casos de pruebas.
- ✓ Incorporación de los casos de pruebas a las herramientas seleccionadas.
- ✓ Ejecución de las pruebas.
- ✓ Elaboración del informe o manual de uso de las herramientas.
- ✓ Determinación de los factores que pueden causar riesgos durante la ejecución de las pruebas.
- ✓ Valoración de los resultados de las pruebas.

Introducción

Luego de delimitar tanto el objetivo general como las tareas de investigación se precisa que la ***idea a defender*** es: la automatización de las pruebas de carga y estrés al proyecto SIPP permitirá verificar los requerimientos de rendimiento de la solución y predecir el comportamiento de la aplicación.

Durante todo el trabajo investigativo se emplearon varios ***métodos científicos de investigación***, que facilitan el estudio de las características del tema en cuestión. Como ***métodos teóricos*** se utilizaron: *Inductivo – Deductivo;* el cual permitió el estudio de casos individuales relacionados con la temática para llegar a una conclusión general de la misma , también se hizo uso del *Analítico- Sintético* con el objetivo de buscar la esencia de los fenómenos, los rasgos que los caracterizan y los distinguen, de igual modo se empleó el *Análisis histórico lógico* permitiendo estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo.

Asimismo se utilizó el método *Análisis de las fuentes de información* para analizar las teorías y documentos posibilitando la extracción de los elementos más importantes que se relacionan con el objeto de estudio, la *Entrevista* para obtener información eficazmente, a través de preguntas a profesionales que dominan la problemática en la que está enmarcada la investigación brindando ideas para profundizarla y la *Encuesta* que se aplicó a líderes de proyectos, para buscar más información y ganar en objetividad de la misma (Ver anexo 1) ; los métodos expuestos anteriormente se clasifican como ***métodos empíricos***.

Como ***método del nivel estadístico*** se empleará la *estadística descriptiva* para ilustrar cuantitativamente los resultados de la comparación entre las herramientas, mediante porcientos y gráficos.

La investigación quedó estructurada en tres capítulos. En el Capítulo 1 se hace una fundamentación teórica, destacando los antecedentes, tecnologías, tendencias, principales funcionalidades, características y haciendo una descripción comparativa entre diferentes herramientas para la automatización de pruebas para aplicaciones web. En el Capítulo 2, para desarrollar la solución se crea el plan de pruebas, el diseño de casos de pruebas, se aplica la estrategia de prueba propuesta al proyecto, se desarrolla un informe o manual de uso de las herramientas. En el Capítulo 3 se realiza la validación de la propuesta a partir de la valoración de expertos en pruebas de software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

Este capítulo evidencia los temas fundamentales en los que se basa la investigación. Se hará referencia a la definición de la prueba de software, sus objetivos, principios, etapas involucradas y ventajas, así como la metodología a emplear durante las pruebas. Se realizará un estudio de las herramientas que permiten automatizar pruebas de carga y estrés, para seleccionar las que se aplicarán al procedimiento.

1.2 Pruebas de software.

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo (1).

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. Es un proceso de ejecución de un programa con la intención de descubrir un error, no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software (2).

Objetivos de las pruebas de software.

La prueba de software es un elemento crítico para la garantía del correcto funcionamiento del software. Entre sus objetivos están:

- ✓ Detectar defectos en el software.
- ✓ Verificar la integración adecuada de los componentes.
- ✓ Verificar que todos los requisitos se han implementado correctamente.
- ✓ Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.

- ✓ Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo (3).

Para lograr los objetivos propuestos, un ingeniero de software deberá conocer los principios básicos que guían las pruebas del software.

Principios de las pruebas de software.

Las pruebas se rigen por una serie de principios, una buena comprensión de estos facilitará el posterior uso de los métodos en un efectivo diseño de casos de prueba. A continuación se citan:

- ✓ La prueba puede ser usada para mostrar la presencia de errores, pero nunca su ausencia.
- ✓ La principal dificultad del proceso de prueba es decidir cuándo parar.
- ✓ Evitar casos de pruebas no planificados, no reusables y triviales a menos que el programa sea verdaderamente sencillo.
- ✓ Una parte necesaria de un caso de prueba es la definición del resultado esperado.
- ✓ Los casos de pruebas tienen que ser escritos no solo para condiciones de entrada válidas y esperadas sino también para condiciones no válidas e inesperadas.
- ✓ El número de errores sin descubrir es directamente proporcional al número de errores descubiertos (4).

Estas leyes que definen básicamente la aplicación de las pruebas de software ayudan a refinar el producto de software a través de las etapas involucradas.

1.2.1 Etapas involucradas en las pruebas de software.

1. Seleccionar qué es lo que debe medir la prueba, es decir, cuál es su objetivo, para qué exactamente se hace la prueba.

2. Decidir cómo se va a realizar la prueba, es decir, qué clase de prueba se va a utilizar para medir la calidad y qué clase de elementos de prueba se deben usar.

3. Desarrollar los casos de prueba. Un caso de prueba es un conjunto de datos o situaciones de prueba que se utilizarán para ejecutar la unidad que se prueba o para revelar algo sobre el atributo de calidad que se está midiendo.

4. Determinar cuáles deberían ser los resultados esperados de los casos de prueba y crear el documento que los contenga.

5. Ejecutar los casos de prueba.

6. Comparar los resultados de la prueba con los resultados esperados. Cualquier discrepancia entre ellos significa un error. Típicamente el error está en el sistema o unidad probada, pero también puede ser generado por algún aspecto del mismo proceso de prueba (5).

Para que cada una de las etapas mencionadas anteriormente se realice de forma eficaz es necesario definir y describir la estrategia de prueba a seguir.

1.2.2 Estrategia de prueba.

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la evaluación correcta del software.

Las características generales son:

- ✓ La prueba comienza en el nivel de módulo y trabaja “hacia afuera”.
- ✓ En diferentes puntos son adecuadas a la vez distintas técnicas de prueba.
- ✓ La prueba y la depuración son actividades diferentes.

La estrategia define:

- ✓ Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
- ✓ Qué criterios de éxito y culminación de la prueba serán usados.
- ✓ Consideraciones especiales afectadas por requerimientos de recursos o que tengan implicaciones en la planificación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Los objetivos de la estrategia de prueba son:

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

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

- ✓ Realizar diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Los productos de desarrollo de software en los que se detectan defectos son probados de nuevo y posiblemente devueltos a otra etapa, como diseño o implementación, de forma que los defectos puedan ser arreglados (3).

Con la realización de una correcta guía para el probador de software o estrategia de pruebas se debe proceder a examinar el software para lo que es necesario algunas recomendaciones de suma importancia.

Recomendaciones para unas pruebas exitosas

- ✓ Cada caso de prueba debe definir el resultado de salida esperado que se comparará con el realmente obtenido.
- ✓ El programador debe evitar probar sus propios programas, ya que desea (consciente o inconscientemente) demostrar que funcionan sin problemas.
- ✓ Se debe inspeccionar a conciencia el resultado de cada prueba, para así, poder descubrir posibles síntomas de defectos.
- ✓ Al generar casos de prueba, se deben incluir tanto datos de entrada válidos como no válidos.
- ✓ Las pruebas deben centrarse en dos objetivos: probar si el software no hace lo que debe hacer o viceversa, es decir, si provoca efectos secundarios adversos.

- ✓ No deben hacerse planes de prueba suponiendo que, prácticamente, no hay defectos en los programas y, por lo tanto, dedicando pocos recursos a las pruebas siempre hay defectos.

- ✓ La experiencia parece indicar que donde hay un defecto hay otros, es decir, la probabilidad de descubrir nuevos defectos en una parte del software es proporcional al número de defectos ya descubierto.
- ✓ Las pruebas son una tarea tanto o más creativa que el desarrollo de software. Siempre se han considerado las pruebas como una tarea destructiva y rutinaria (6).

Siguiendo rigurosamente estas recomendaciones para ejecutar pruebas de software sin dudas se obtendrán las ventajas que se citan en el próximo epígrafe.

1.2.3 Ventajas de probar el software.

Las pruebas se integran dentro de las diferentes fases del ciclo del software para detectar defectos, permitiendo evaluar el grado de cumplimiento respecto de las especificaciones iniciales del sistema por lo que realizarlas a tiempo proporciona numerosas ventajas:

- ✓ Permite detectar errores en las fases tempranas del proceso de desarrollo, lo que reduce que el costo de corrección.

- ✓ Comprueban la compatibilidad y funcionalidad de las interfaces entre las distintas partes que componen un sistema, estas pueden ser módulos, aplicaciones individuales, aplicaciones cliente/servidor.

- ✓ Determinan hasta donde puede soportar el programa determinadas condiciones extremas.

- ✓ Verifican la calidad del software con respecto a los requerimientos basados en la confiabilidad, funcionalidad, desempeño de la aplicación y del sistema.

- ✓ Validan las funciones del producto de software y que los requerimientos hayan sido implementados apropiadamente.

- ✓ Identifican los defectos del software y permiten corregirlos antes de su instalación.
- ✓ Las pruebas con buena documentación son concisas y realmente fáciles de entender.
- ✓ Permiten realizar reimplementación debido a fallos de seguridad, rendimiento, o simplemente porque el software no reunía lo que el cliente esperaba de éste, permitiendo así que se vuelva a desarrollar de manera segura acorde con la especificación (7).

La necesidad de hacer las pruebas se evidencia en que son enormes las posibilidades de que se cometan errores por los desarrolladores, dada la imposibilidad de trabajar y comunicarse de manera perfecta; pero empleando una metodología adecuada se pueden detectar desde etapas muy tempranas.

1.3 Metodologías de Desarrollo de Software.

Disponer de metodologías diferentes para aplicar a proyectos que se desarrollen resulta imprescindible, de acuerdo a las necesidades cambiantes que tiene el entorno de desarrollo actual y el acelerado progreso de la informática a nivel mundial. Estas metodologías se clasifican en tradicionales y ágiles.

1.3.1 Metodologías tradicionales.

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Por lo que enfatizan en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto de software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada.

1.3.2 Metodologías ágiles.

Las metodologías ágiles surgen como respuesta a problemas reales Se basan en el sentido común, pero rompen con creencias arraigadas. Se están extendiendo con rapidez y se caracterizan por la efectividad que muestran en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad (8).

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

A pesar de que las metodologías ágiles ofrecen un proceso más sencillo que las tradicionales, estas últimas brindan una documentación muy completa, exhaustiva, además de un plan de proyecto cuidadosamente definido. Su desarrollo se basa en un modelo de procesos estrictamente determinado y también está provista de un alto grado de ordenamiento. Una de las metodologías que cumple estas características favorables es el Proceso Unificado del Desarrollo de Software (conocido como RUP, por sus siglas en inglés) que es la que integra el Lenguaje de Modelado Unificado (conocido como UML, por sus siglas en inglés), provee de un proceso de prueba bien definido, en el que se precisa un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo.

1.3.3 Proceso Unificado del Desarrollo de Software.

Los aspectos distintivos del Proceso Unificado están capturados en tres conceptos clave: Dirigido por casos de uso, Centrado en la arquitectura, Iterativo e incremental.

Está integrado por nueve flujos de trabajo seis de ingeniería y tres de apoyo. El Flujo de Trabajo de Prueba es el encargado de evaluar la calidad del producto que se desarrolla, pero no para aceptarlo o rechazarlo al final del ciclo de vida, sino que debe ir integrado al mismo.

El Proceso Unificado está compuesto de cuatro fases, específicamente en este flujo se comportan de la siguiente forma: ***Inicio*** que es donde se hace parte de la planificación de las pruebas definiendo el ámbito del sistema; ***Elaboración*** que es donde se centra la realización de las pruebas cuando se prueba la línea base ejecutable de la arquitectura así como en ***Construcción*** cuando el grueso del sistema esté implementado y durante la ***Transición*** el centro se desplaza hacia la corrección de defectos durante los primeros usos y a las pruebas de regresión. Se basa en componentes, lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas, utiliza el Lenguaje de Modelado Unificado (UML) en la preparación de todos los planos del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

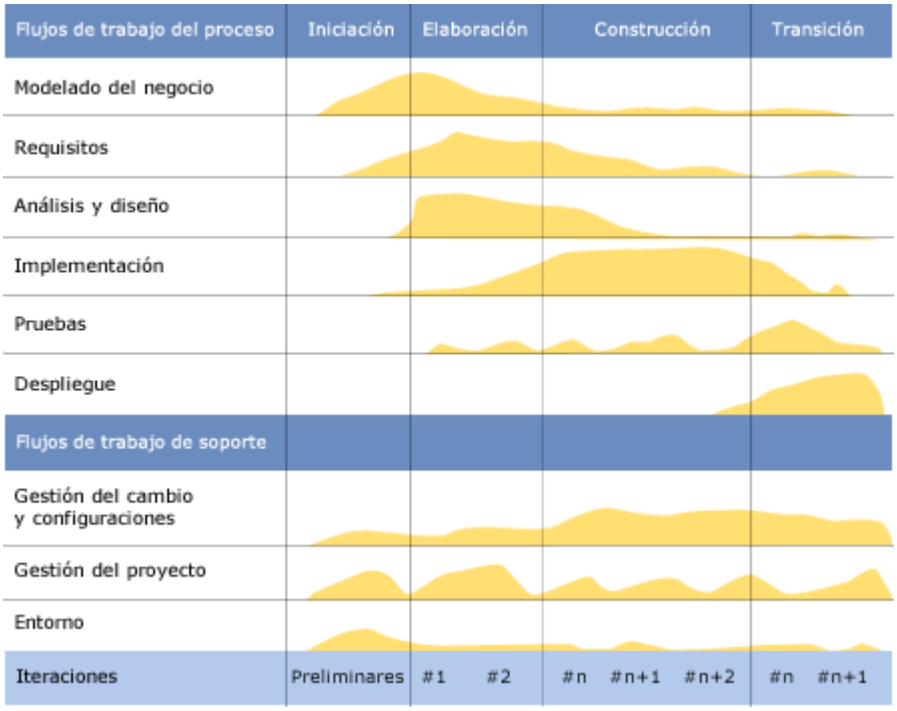


Figura 1 Fases, Flujo de trabajo de RUP.

RUP ofrece diferentes ventajas, entre ellas se destacan: sus casos de uso facilitan las tareas de programación y el posterior diseño de casos de prueba, su enfoque iterativo reduce el riesgo a los costes de un solo incremento, disminuye los retrasos en el calendario y acelera el desarrollo, por lo que los trabajadores se desempeñan de manera más eficiente al obtener resultados a corto plazo, permite un entendimiento incremental del problema a través de refinamientos sucesivos ,habilita una fácil retroalimentación de usuario, permite que el equipo de desarrollo mantenga su atención en producir resultados; el progreso es medido conforme avanzan las implementaciones con lo que se logra reducir los riesgos del proyecto y tener un subsistema ejecutable tempranamente, además RUP permite el seguimiento detallado en cada una de las fases.

Los objetivos de las pruebas definidos en el Flujo de Trabajo Prueba descrito por RUP se especifican a continuación:

- ✓ Planificar las pruebas necesarias para cada iteración.
- ✓ Diseñar e implementar las pruebas creando los casos de prueba que especifican que probar.
- ✓ Realizar las diferentes pruebas evaluando los resultados periódicamente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Los trabajadores definidos por RUP en el Flujo de Trabajo Prueba son:

- ✓ Diseñador de pruebas: Planifica las pruebas. Es responsable de la integridad del modelo de pruebas asegurando que el modelo cumple con su propósito. Identifica, prioriza y describe los casos de prueba y los procedimientos correspondientes, reconociendo las técnicas apropiadas, herramientas y prototipos para consolidar las pruebas y define además la configuración del entorno para realizar las pruebas.

- ✓ Ingeniero de componentes: Es responsable de los componentes de pruebas que automatizan algunos de los procedimientos de pruebas.

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

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

RUP define 4 Niveles de Pruebas:

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas:

Prueba de Unidad: Es la prueba enfocada a los elementos más pequeños del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos y que ellos funcionen como se espera.

Prueba de Integración: Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores o incompletitud en las especificaciones de las interfaces de los paquetes.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Prueba de Sistema: Son las pruebas que se hacen cuando el software está funcionando como un todo.

Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.

En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

El objetivo de las pruebas de sistema es la comprobación del sistema global, realizándose para comparar el sistema con sus objetivos originales.

- ✓ Se busca demostrar qué objetivos no se cumplen.
- ✓ No es posible realizarlo si no existen objetivos documentados.
- ✓ No significa probar la funcionalidad en su conjunto.
- ✓ Se usan como base los objetivos originales y la documentación del usuario para ampliarla.
- ✓ No existe un método, se dan lineamientos, a la hora de preparar los casos de prueba.
- ✓ Test de Sistema: Se finaliza cuando se cumplieron los meses programados y se han hallado errores.

Tipos de Pruebas del Sistema:

- ✓ Prueba de Recuperación: Es una prueba del sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente.
- ✓ Prueba de Seguridad: Intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de acceso impropio.
- ✓ Prueba de Resistencia**:** Están diseñadas para enfrentar a los programas con situaciones anormales.
- ✓ Prueba de Rendimiento**:** Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado (9).

  Los tipos de pruebas de rendimiento que habitualmente pueden ponerse en marcha son los siguientes:

  - ***Prueba normal.*** Permite establecer los tiempos medios de respuesta cuando sólo un usuario está conectado a la aplicación. Esta prueba pretende establecer una

referencia futura para posteriores comparaciones así como medir unitariamente el software entregado.

- ***Prueba con número mínimo de usuarios.*** Se realizan las pruebas del sistema con el número de usuarios mínimos concurrentes establecido.
- ***Prueba con número máximo de usuarios.*** Se realizan las pruebas del sistema con el número de usuarios máximo concurrentes establecido.
- ***Prueba de número máximo soportado de usuarios.*** Se busca encontrar cuál es el límite del sistema (10).

Con el objetivo de evaluar el rendimiento del sistema se realizan tanto las pruebas de carga como las de estrés.

Una **prueba de carga** se ejecuta para comprender el comportamiento de una aplicación ante una carga determinada, puede ser el número de usuarios esperado ejecutando una cifra de transacciones durante un tiempo determinado. El resultado nos dará el tiempo de respuesta de todas las transacciones críticas. Si la base de datos, servidor de aplicación también se monitorizan, entonces esta prueba puede mostrar potenciales problemas en la aplicación (11).

En el siguiente fragmento Boris Beizer[1] describe de forma muy clara que significa prueba de carga:

"El Test de Carga somete a un sistema a una carga estadísticamente representativa. [...] En Test de Performance, la carga se varía de un mínimo (cero) hasta el nivel máximo que el sistema puede sostener sin quedar sin recursos o tener transacciones sufriendo (dependiendo de la aplicación) demoras excesivas. Otro uso del término de Test de Carga es como una prueba cuyo objetivo es determinar la carga máxima sostenible que el sistema puede manejar. En este uso, prueba de carga es meramente comprobar con las tasas más altas de llegada de transacciones."

Es importante destacar que en una prueba de carga el sistema debe responder al nivel de carga sometido de forma adecuada, en otras palabras según sus requerimientos. Si el sistema no responde de forma esperada, los resultados obtenidos no corresponden a una prueba de carga, sino a una de estrés (12).

[1] Boris Beizer es un ingeniero de software y escritor estadounidense. Ha escrito numerosos libros y artículos sobre temas como la arquitectura del sistema y pruebas de software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Las **pruebas de estrés** son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento, mediante la ejecución de un número de usuarios muy superior al esperado. Esta prueba tiene como finalidad determinar la robustez de una aplicación cuando la carga es extrema y ayuda a administradores a determinar si la aplicación se comportará correctamente en dichas situaciones (11).

La prueba de estrés determina el punto en que un sistema comienza a brindar un desempeño inaceptable. Este punto puede ser llamado punto de quiebre. La idea es estresar a un sistema al punto de quiebre para encontrar errores que podrían ser potencialmente perjudiciales. Esta es la única forma de poder detectar este tipo de errores. No se espera que el sistema procese la sobrecarga sin los recursos adecuados, pero sí que se comporte de una manera razonable.

Boris Beizer define este concepto de la siguiente manera:
"Test de estrés es someter a un sistema a una carga no razonable mientras se le niegan los recursos (por ejemplo, memoria RAM, disco, mips, interrupciones, entre otros) necesarios para procesar esa carga (12). "

Prueba de Aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (9).

RUP propone diferentes Tipos de pruebas. Cada tipo de prueba tiene un objetivo específico y una técnica que lo soporte. (13)

Tipo de prueba	Objetivo
Funcionalidad	**Función:** Pruebas fijando su atención en la validación de las funciones, métodos, servicios, caso de uso.
	Seguridad: Asegurar que los datos o el sistema solamente es accedido por los actores deseados.
	Volumen: Enfocada en verificar las habilidades de los programas para manejar grandes cantidades de datos, tanto como entrada, salida o residente en la BD.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Usabilidad	**Usabilidad:** Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.
Fiabilidad	**Integridad:** Enfocada a la valoración de la robustez (resistencia a fallos).
	Estructura: Enfocada a la valoración a la adherencia a su diseño y formación. Este tipo de prueba es hecho a las aplicaciones Web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado.
	Benchmark: Es un tipo de prueba que compara el rendimiento de un elemento nuevo o desconocido a uno de carga de trabajo de referencia conocido.
	Contención: Enfocada a la validación de las habilidades del elemento a probar para manejar aceptablemente la demanda de múltiples actores sobre un mismo recurso.
Rendimiento	**Carga:** Usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales.
	Estrés: Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponibles)
	Performance profile: Enfocadas a monitorear el tiempo en

	flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccionar los cuellos de botellas y los procesos ineficientes.
Soporte	**Configuración:** Enfocada a asegurar que funciona en diferentes configuraciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema.
	Instalación: Enfocada a asegurar la instalación en diferentes configuraciones de hardware y software bajo diferentes condiciones.

Tabla 1 Tipos de pruebas propuestas por RUP.

Artefactos generados en el Flujo de Trabajo de Prueba propuesto por RUP:

En el Flujo de Trabajo de Prueba propuesto por RUP se generan diferentes artefactos con tareas específicas. Se pueden citar entre ellos: ***Caso de prueba*** que especifica una forma de comprobar el sistema, incluyendo las entradas, resultados y las condiciones bajo las que ha de probarse, ***Procedimiento de Prueba*** encargado de especificar cómo realizar uno o varios casos de prueba o partes de estos, también el ***Componente de prueba*** que automatiza uno o varios procedimientos de prueba o partes, asimismo el ***Plan de Pruebas*** que describe las estrategias, recursos y planificación de la prueba, así como la ***Evaluación de prueba*** donde se valoran los resultados de las mismas. Todos estos artefactos se generarán durante el proceso de pruebas automatizadas de carga y estrés (14).

1.4 Automatización de pruebas.

Las pruebas automatizadas constituyen una necesidad casi absoluta en los proyectos que impliquen la creación y desarrollo de software, estas incluyen una amplia gama de beneficios que no están disponibles para las pruebas manuales.

En la actualidad existe una gran variedad de herramientas para realizar pruebas automatizadas. Estas proporcionan una ventaja evidente partiendo de tiempo y conservación de los recursos, sin comprometer la calidad del procedimiento de prueba, la exactitud de los informes finales y la

eficiencia del proceso de pruebas, son efectivas y económicamente factibles. El uso de las mismas permite tener más comodidad en la elaboración de los errores y fallos que se producen antes de que el producto de software es liberado, o en momentos en que hay una necesidad de proporcionar asistencia al cliente para resolver los defectos encontrados.

Ventajas de las pruebas automatizadas.

Las pruebas automatizadas han mejorado enormemente el proceso básico de perfeccionamiento de aplicaciones web y sus beneficios se encuentran disponibles para desarrolladores. A continuación se citan algunas de sus ventajas:

- ✓ Confiables: Las pruebas realizan con exactitud diversas operaciones cada vez que se ejecutan, de tal modo que evita posibles errores humanos.
- ✓ Recursivas: Evidencian cómo el software reacciona bajo diferentes condiciones, cuando se está probando una misma operación repetidamente.
- ✓ Programables: Permiten programar pruebas sofisticadas que pongan en evidencia la robustez del software que se pruebe.
- ✓ Abarcadoras: Facilitan la construcción de un ambiente de pruebas que cubra cada característica del software.
- ✓ Reutilizables: Permite reutilizar pruebas en diversas versiones.
- ✓ Factibles: Se pueden ejecutar más pruebas en menos tiempo y el número de los recursos se reduce.
- ✓ Rápidas: Su ejecución es perceptiblemente más rápida.
- ✓ Flexibles: Las pruebas deben ser fáciles de entender, de modificar y de extender (15).

Las pruebas automatizadas se ejecutan mediante el empleo de herramientas que facilitan las tareas de un probador de software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.4.1 Herramientas para pruebas automatizadas.

Las herramientas de pruebas automatizadas permiten crear un “marco de ejecución de pruebas automatizadas” lo que permite leer de una fuente de datos externa que contenga los casos de negocio y sumarlo a los casos de pruebas, para adaptarlos a la aplicación. El marco de pruebas automatizado debe ser capaz de analizar, si luego de introducir todos los datos leídos, el resultado que muestra el sistema es el esperado; el marco de pruebas será capaz de adaptar a la aplicación, los casos de pruebas, teniendo como entrada una fuente de datos dinámica (16).

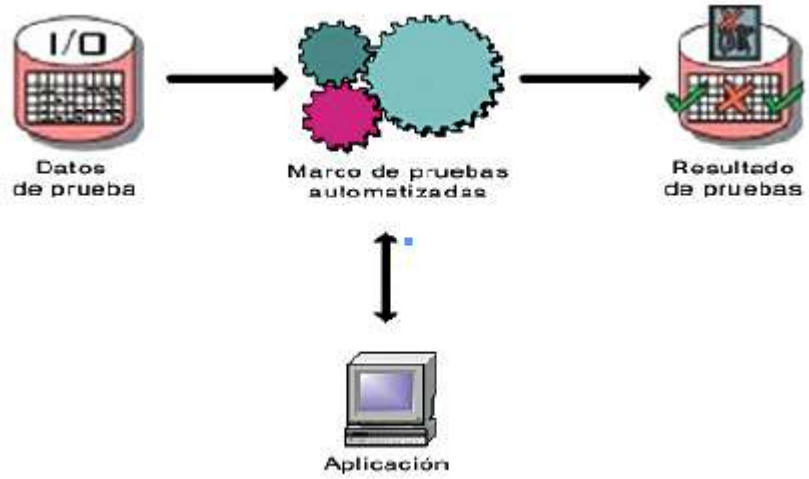


Figura 2 Marco de pruebas automatizadas.

1.4.1.1 Pruebas automatizadas en la UCI.

La Universidad ha tenido experiencias en el uso de herramientas para la automatización de pruebas, el ejemplo más significativo es en el Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos (Calisoft).Este centro ante la liberación del producto “Proyecto de Modernización del Cuerpo de Investigaciones Científicas, Penales y Criminalística de la República Bolivariana de Venezuela” (CICPC) lo expuso a una rigurosa etapa de pruebas de carga y estrés sometiendo la aplicación a situaciones de concurrencia y de flujo de datos extremos, haciendo uso de la herramienta automatizada JMeter sobre Windows como Sistema Operativo. Esta es una herramienta que permite realizar pruebas funcionales y de rendimiento sobre aplicaciones web, comprueba el funcionamiento y el rendimiento de diversos recursos de

software, se destaca por su versatilidad, estabilidad y por ser de uso gratuito. Las pruebas valieron para determinar en qué medida el producto cumplió con las expectativas del cliente y permitieron detectar anomalías que incurrían en el rendimiento del sistema, el tiempo de respuesta del mismo y la cantidad de usuarios que soportaba la aplicación.

En el presente epígrafe se dará una breve explicación de las herramientas para pruebas automatizadas que se tuvieron en cuenta en la investigación para elegir las más adecuadas para pruebas de carga y estrés.

1.4.1.2 Herramientas para pruebas de carga.

VPerformer

Es un producto de pruebas de rendimiento y de carga, que puede ser usado para evaluar el rendimiento y escalabilidad de las aplicaciones web. Revela el comportamiento de la aplicación web cuando está sometido a una tensión extrema.

Posibilita medir las características de rendimiento de la aplicación mediante la generación de scripts de pruebas automatizadas y reproducir scripts de prueba que simulan miles de usuarios virtuales concurrentes.

Permite monitorear el estado de cualquier componente externo asociado que influya en el comportamiento de su aplicación, permitiéndole identificar los cuellos de botella. Entre sus principales beneficios se destacan: una interfaz fácil de usar, no requiere conocimientos de programación y aporta flexibilidad a las pruebas de distribución (17).

WebLoad Professional

Está específicamente diseñado para las pruebas de carga de aplicaciones de Internet simulando el tráfico de hasta decenas de miles de usuarios y ofreciendo información detallada en función de la carga. Se basa en estándares abiertos y utiliza JavaScript como lenguaje de secuencias de comandos de edición.

Entre sus principales características se destacan: la flexibilidad con respecto a la concesión de licencias para ofrecer un coste total de propiedad reducido notablemente, permite la reutilización de scripts de prueba, sus soluciones y complementos incluyen soporte para aplicaciones

dinámicas de Internet con el Ajax y Flex de Adobe, Web Services, ERP / CRM Applications, aplicaciones de Business Intelligence, protocolos de legado y más (18).

OpenLoad Tester

Es la primera solución rápida de optimización de rendimiento basada en navegador. Fácil de usar, para las pruebas de carga y estrés de aplicaciones. Impulsado por el servidor IBM WebSphere y DB2 base de datos universal y completamente integrado con el desarrollo de aplicaciones de IBM WebSphere Studio.

El verificador de carga OpenLoad Tester brinda facilidad de uso, exactitud y escalabilidad de un solo desarrollo integrado y ambiente de la prueba. Este minimiza sustancialmente el tiempo requerido para las pruebas de estrés y pone a punto la eficiencia de la web basado en aplicaciones y servicios por simplificación de procesos, verificando la conducta funcional esperada a través de la regresión automatizada (19).

Tiene poderosas capacidades de ***pruebas de regresión*** incorporada que le permiten descubrir las causas de nuevos errores, carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, inducidos por cambios recientemente realizados en partes de la aplicación que antes del citado cambio no eran propensas a este tipo de error. Esto implica que el error tratado se reproduce como consecuencia inesperada del cambio en el programa (20).
Tiene un controlador modelo de agente, que le permite distribuir la carga entre todas las máquinas agentes múltiples; permitiendo llevar a cabo una misión de manera autónoma e inteligente, así como de pruebas desde diferentes puntos en la red (19).

1.4.1.3 Herramientas para pruebas de estrés.

Webserver Stress Tool

Simula grandes números de usuarios accediendo un sitio web vía HTTP/HTTPS. Basado en los parámetros que especifica, la aplicación no sólo pide el HTML de una URL, sino también frames, imágenes, archivos de Flash, emulando el mismo comportamiento que un navegador web debería mostrar al acceder al sitio web. Funciona con todos los servidores web, con soporte total para SSL.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Entre sus principales beneficios se destacan: permitir patrones de URL simples, así como patrones complejos, a través de un archivo de Script, asegurar que los incidentes críticos en su sitio web se resuelvan antes que tiren abajo sus recursos web y certificar que sus sitios web y aplicaciones reciben los recursos de servidor que necesitan para garantizar una experiencia de usuario de gran calidad y que está aprovechando al máximo la inversión de su tecnología de servidor a través de consistentes análisis y pruebas en profundidad(21).

Wapt

Es una herramienta de prueba de carga y estrés que está diseñada para minimizar la curva de aprendizaje y dar al usuario la habilidad de crear una pesada carga de trabajo. Utiliza una serie configurable de usuarios virtuales para simular una carga de la vida real. Se soportan pruebas de aplicaciones web dinámicas, grabación y reproducción de peticiones HTTPS y varios sistemas de autenticación.

Entre sus principales funcionalidades resaltan que: el proceso de instalación es muy simple, pero su uso indica la aceptación de los términos y condiciones establecidos en la licencia, proporciona además resultados de la prueba informativa a través de gráficos e informes descriptivos, pudiendo probar y analizar las características de rendimiento bajo condiciones de carga diferentes para encontrar los cuellos de botella de sus aplicaciones web (22).

Hammerhead

Herramienta de prueba de estrés diseñada para probar desde fuera servidores y sitos web. Ha sido diseñada para emular varios usuarios conectados desde múltiples direcciones IP.

Es altamente configurable y admite peticiones basadas en escenario. Tiene muchas opciones para generar distintos problemas a sitios web. Disponible para plataformas Linux, Solaris y FreeBSD. En el sistema verifica el rendimiento operativo en las condiciones en tiempo real.

Las cargas que ocasiona Hammerhead son un conjunto de solicitudes de un número de archivos, cada uno de los cuales puede contener un número de escenarios (una solicitud por escenario). Los escenarios pueden estar vinculados en secuencias, a fin de simular acciones de los usuarios reales. Una vez que los escenarios están cargados, Hammerhead pone en marcha una serie de hilos y envía solicitudes al puerto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El resultado esperado de una solicitud que se especifique en un escenario y cualquier resultado desde el puerto que no se corresponde con el resultado previsto será reportado como un error. Si no se especifican los resultados esperados, entonces cualquier resultado desde el puerto será aceptado como válido. La especificación de un resultado esperado para cualquier escenario tiene el efecto de especificar el mismo resultado para todos los escenarios que tienen la misma petición. Cualquier imposibilidad de tener una conexión con el puerto, o cualquier hecho de no recibir una respuesta a la solicitud se indicará también.

Con esta herramienta se puede mejorar la experiencia del cliente que interactúa con el producto e identificar y corregir problemas a tiempo (23).

Teniendo en cuenta las características de las herramientas estudiadas se realizó una comparación de las mismas para seleccionar las más idóneas a aplicar al proyecto SIPP. Para realizar una correcta elección de estas es necesario hacer un balance importante entre las candidatas, enfocado a algunos de sus aspectos. A continuación se destacan los seleccionados:

Dirección URL del fabricante: Sitio web en el cual se encuentran los manuales, explicaciones y archivos instaladores de la herramienta.

Tipo de licencia y precio: El tipo de licencia delimita la forma de uso y está directamente relacionado con el precio, ya que dependiendo de las capacidades o límites del aplicativo los precios suben o bajan según sea el caso. Generalmente en éste tipo de aplicaciones el precio lo delimita el número máximo de usuarios concurrentes que es posible emular.

Plataforma: Sistemas soportados para la instalación de la herramienta que en términos precisos se traduce en los sistemas operativos soportados o el lenguaje de programación usado para la construcción de la herramienta que en algunos casos se traduce en portabilidad entre sistemas, tal caso son las herramientas programadas en Java, las cuales generalmente son portables entre sistemas Windows, Linux, UNIX, Solaris, entre otros.

Requisitos de hardware*:* Requisitos del sistema a nivel de hardware que se requieren para que la herramienta se ejecute correctamente. Se refiere generalmente al tipo de procesador, memoria RAM y disco duro.

Beneficios: Ventajas que ofrece la herramienta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

La representación de estos aspectos comparativos se muestra en las tablas siguientes:

Herramientas de carga	**Dirección URL del fabricante**	**Tipo de licencia y precio**	**Plataforma**	**Requisitos de hardware**	**Beneficios**
vPERFORMER	http://www.verisium.com	Licencia Flotante $995.00	Windows XP/2000/2003/NT	3 sistemas independientes pero no especificadas sus características	Interfaz fácil de usar. No requiere conocimientos de programación. Flexibilidad de las pruebas de distribución.

WEBLOAD PROFESSIONAl	http://www.radview.com/product/Adons-RM.aspx	Juicio libre	Windows 2000 Professional, XP Pro, Server 2003	**CPU:** Pentium III 800 MHz, **RAM:** mínimo 512MB, pero recomendado 1GB. 303.88 MB de **espacio en disco**.	Flexibilidad con respecto a la licencia. Permite la reutilización de scripts de prueba. Sus soluciones y complementos incluyen soporte para aplicaciones dinámicas de Internet.
OPENLOAD TESTER	http://www.opendemand.com/openload/faq.shtml	GPL - GNU Licencia Pública	Multiplataforma	**CPU:** Pentium IV 1 GHz, **RAM:**512MB, 1 MB de **espacio libre en disco**.	Facilidad de uso, exactitud y escalabilidad. Minimiza el tiempo requerido para las pruebas de estrés. Poderosas capacidades de pruebas de regresión. Permite aprovechar los escenarios de carga de prueba existentes.

Tabla 2 Comparación de las herramientas de carga.

A pesar de que todas las herramientas expuestas ofrecen variadas funcionalidades y proporcionan ventajas con su empleo, la herramienta OpenLoad Tester ha demostrado ser la forma más fácil de optimizar el rendimiento de las aplicaciones y servicios web, aumentando el nivel de escalabilidad al aprovechar la potencia combinada para generar usuarios por máquina virtual más que cualquier otra herramienta de pruebas de tensión. Tiene abundante documentación y es de licencia pública.

Herramientas de estrés	Dirección URL del fabricante	Tipo de licencia y precio	Plataforma	Requisitos de hardware	Beneficios
WEBSERVER STRESS TOOL	http://www.paessler.com/webstress	$249.95	Windows XP/2000/2003/2008/Vista/7	**CPU:** 1GHz, **RAM:** 512 MB, 5-10 MB de **espacio en disco,** además de espacio para los archivos de registro (puede ser de varios MB) y alrededor de 100 KB de RAM por cada usuario simulado.	Permite patrones de URL simples, así como patrones complejos, a través de un archivo de Script. Asegura que los incidentes críticos en un sitio web se resuelvan antes que tiren abajo sus recursos web. Certifica que sus sitios web y aplicaciones reciben los recursos de servidor que necesitan.
WAPT	http://www.loadtestingtool.com/download.shtml	$250	Windows 2000/XP/2003/Vista/2008/Win7	**CPU:** Pentium 4.2GHz, **RAM:** Al menos 512 MB, 5 MB de **espacio libre en disco** para archivos de programa y 50-500 MB de espacio libre en disco para los registros de los	Proceso de instalación es muy simple. Proporciona resultados de la prueba informativa a través de gráficos e informes descriptivos. Analizar las características de rendimiento bajo condiciones de carga diferentes para encontrar los cuellos de botella.

				usuarios virtuales	
HAMMERHEAD	http://linux.die.net/man/1/hammerhead	GPL - GNU Licencia Pública	Multiplataforma	**CPU:** 1.5 GHz, **RAM:** Al menos 512 MB, 2 MB de **espacio libre en disco**	Es altamente configurable y admite peticiones basadas en escenario. Tiene muchas opciones para generar distintos problemas a sitios Web. En el sistema verifica el rendimiento operativo en las condiciones en tiempo real.

Tabla 3 Comparación de las herramientas de estrés.

Con la comparación realizada anteriormente y por la superioridad evidenciada en la herramienta Hammerhead en cuanto a los aspectos analizados, se comprobó que es totalmente configurable y que contiene muchas otras opciones para someter un sitio a condiciones anormales, el cual siendo expuesto a una rigurosa evaluación demostrará finalmente cuán eficiente puede ser.

Luego de elegir las herramientas OpenLoad y Hammerhead, en el siguiente epígrafe se explica brevemente el surgimiento y objetivos del proyecto al que se le harán las pruebas de carga y estrés con las herramientas seleccionadas.

1.5 Proyecto SIPP.

El Sistema de Información de Perforación de Pozos Petroleros (SIPP) nace para dar respuestas a las necesidades de la industria de perforación petrolera, específicamente en la Isla de Cuba con el objetivo de controlar y asegurar el flujo de información que se genera en los pozos en perforación. Es un sistema distribuido, esto permite tener parte del sistema en el Pozo y a la otra en la Dirección de Intervención y Perforación de Pozos (DIPP) haciendo el sistema más ligero y seguro, el subsistema pozo trabaja en su propio entorno local, utiliza la replicación de datos como solución para comunicar a los pozos con la DIPP, es multiplataforma y de fácil configuración, si

bien es un software a medida del cliente en este caso CUPET, por sus características puede ser reestructurado para operar en negocios con tipos de necesidades específicas del cliente.

El sistema permite visualizar la información en la aplicación a manera de reportes y partes de la perforación, permite hacer representaciones gráficas, así como exportar/importar archivos (24).

Conclusiones Parciales.

En este capítulo se han puntualizado conceptos fundamentales relacionados con la investigación, argumentando la metodología a utilizar enfocada principalmente al tema de pruebas específicamente a las de sistema, se explican brevemente algunas herramientas que se utilizan para realizar pruebas de carga y estrés con sus respectivos beneficios, lo que arrojó convincentes argumentos para realizar una elección basada en cada una de sus características.

El estudio de estas herramientas demostró que las mejores candidatas para aplicar en el proyecto SIPP son Hammerhead y OpenLoad pues entre sus funcionalidades reportan: evaluar el rendimiento del software, evaluar su robustez e identificar fallos o ausencia de ciertos aspectos básicos que deba cumplir la aplicación. A través de dichas funcionalidades y mediante una estrategia de pruebas que será descrita en el Capítulo 2, se realizará un procedimiento que evidencie los pasos a seguir para realizar las pruebas de rendimiento al proyecto SIPP específicamente las de carga y estrés.

CAPÍTULO 2: PROCEDIMIENTO

CAPÍTULO 2: PROCESO AUTOMATIZADO PARA PRUEBAS DE CARGA Y ESTRÉS EMPLEANDO OPENLOAD Y HAMMERHEAD.

2.1 Introducción.

En este capítulo se describe el proceso de pruebas de sistema diseñado para ser aplicado al proyecto SIPP. Se define el plan de pruebas que incluye las estrategias, recursos y planificación de prueba; se puntualizan los casos y procedimientos de pruebas. Por último, se realiza la implementación y ejecución de las pruebas de carga y estrés, culminando el proceso con la evaluación de los resultados.

Breve explicación del Sistema a probar.

El sistema a probar elimina las deficiencias de plataforma. El uso de framework hace más ágil su desarrollo. Posee altas ventajas de mantenimiento, despliegue y soporte. El 80% de las tecnologías usadas son productos de comunidades de desarrollo de software libre. SIPP contiene además otras funciones que lo convierten en un producto único: comunicación entre los usuarios a través de la aplicación, varios idiomas (Español, Inglés y cualquier idioma que se solicite), interfaces configurables, además de permitir al usuario crear nuevos reportes.

2.2. Estrategia para la realización de pruebas de carga y estrés.

La estrategia para la realización de pruebas de carga y estrés proporciona un plano o guía para el probador del software y para la organización de control de calidad. Describe los pasos a llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos y cuánto esfuerzo, tiempo y recursos se van a requerir. Por lo tanto, desarrolla etapas como la planificación de las pruebas, el diseño de los casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes.

2.2.1. Alcance.

La realización de las pruebas de carga y estrés son de suma importancia, cuando se habla de garantizar la calidad del software, en cuanto a la medición del tiempo de respuesta. Se espera que el desarrollo de esta investigación tenga repercusión en el proyecto SIPP y más adelante a nivel de facultad y de universidad.

CAPÍTULO 2: PROCEDIMIENTO

2.2.2 Roles identificados.

La conformación del equipo de trabajo para la correcta aplicación del procedimiento es de suma importancia para su ejecución eficiente. El mismo estará conformado por los siguientes miembros:

Rol	Descripción
Diseñador de Pruebas	Identifica, prioriza, selecciona y describe los casos de pruebas y los procedimientos correspondientes, reconociendo las técnicas apropiadas, herramientas y prototipos para consolidar las pruebas y define además la configuración del entorno para realizar las pruebas.
Ingeniero de componentes	Responsable de los componentes de pruebas que automatizan algunos de los procedimientos de pruebas.
Ingeniero de pruebas de sistema	Realiza las pruebas de sistema sobre los ejecutables y documenta los defectos en las pruebas de sistema.

Tabla 4 Roles para el equipo de pruebas.

2.2.3 Objetivos.

- ✓ Verificar que el sistema esté ajustado para soportar la máxima carga de trabajo.
- ✓ Verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas en correspondencia con los requerimientos.
- ✓ Asegurar que, ante una carga de trabajo determinada, las páginas a las que se acceden respondan en el intervalo de tiempo especificado.
- ✓ Determinar el tiempo medio de respuesta en segundos que obtendrá el usuario.

- ✓ Determinar el número solicitudes realizadas durante el tiempo especificado.
- ✓ Determinar el tiempo de respuesta más alto durante la prueba.
- ✓ Determinar el número de solicitudes completadas.
- ✓ Determinar el número máximo de usuarios concurrentes que pueden acceder a una página específica, o transacciones por segundo que la aplicación es capaz de soportar.
- ✓ Identificar las páginas que responden con mayor y menor rapidez.
- ✓ Poner la aplicación bajo situaciones anormales
- ✓ Determinar el porcentaje de errores.

2.2.4. Fases a desarrollar.

El procedimiento propuesto para su aplicación en el proyecto SIPP, está compuesto por una estrategia que incluye alcance, roles, objetivos y fases en las cuales se desarrollan actividades como se muestra en la siguiente figura:

Figura 3 Estructura del Procedimiento.

CAPÍTULO 2: PROCEDIMIENTO

2.2.4.1. Planificación.

En esta primera fase, para lograr una correcta organización de las tareas, se necesita una planificación que incluya la mayor cantidad de aspectos relacionados con la preparación que lleva la creación de un correcto procedimiento de pruebas. Esto permitirá el éxito del trabajo realizado. El proyecto SIPP cuenta con un cronograma para llevar a cabo el proceso de pruebas al software. Una vez concluida la implementación del mismo, se inician las pruebas internas. Para ello el equipo encargado de aplicar las pruebas, en correspondencia al cronograma y de mutuo acuerdo con el líder del proyecto, recibe la aplicación a probar .El jefe del equipo en conjunto con sus miembros pasan a la segunda etapa, donde se realizan las siguientes actividades:

- ✓ Estudio de la documentación.
- ✓ Definir y analizar casos de uso a probar.
- ✓ Diseño del entorno de pruebas.

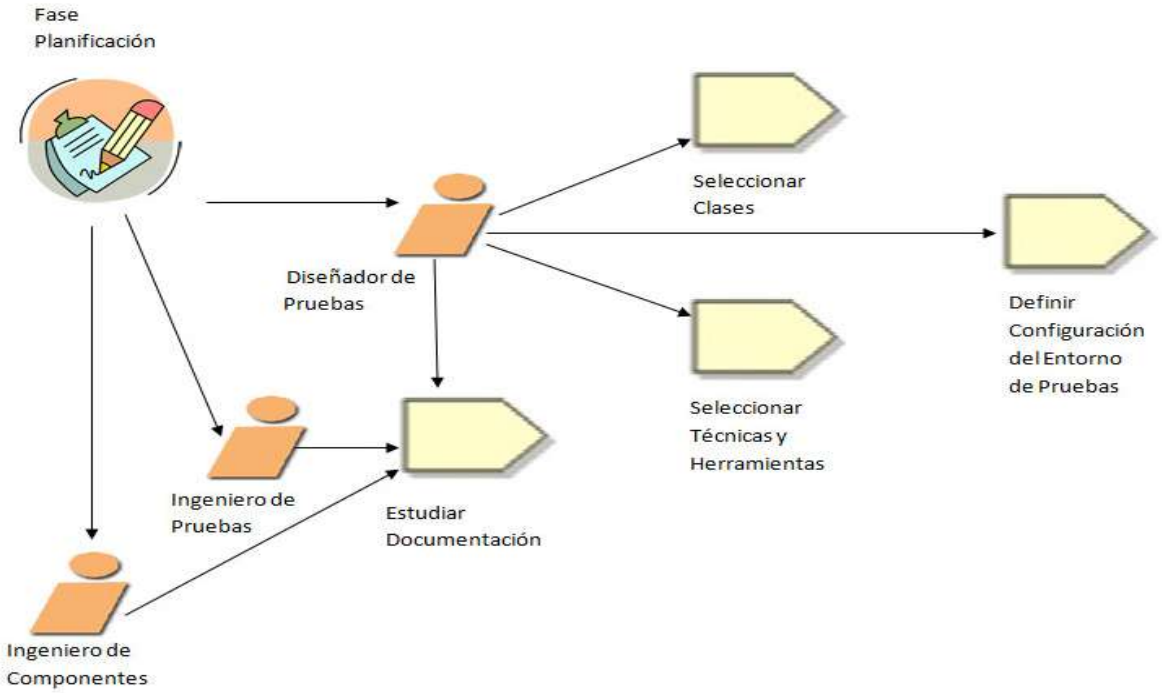


Figura 4 Actividades por roles Fase de planificación.

CAPÍTULO 2: PROCEDIMIENTO

2.2.4.1.1. Estudio de la documentación.

Las funcionalidades y casos de uso del proyecto SIPP deben de estar correctamente detallados en los documentos Especificación de Requisitos de Software y Especificación de Casos de Uso respectivamente, estos documentos junto a Glosario de Términos, el Manual de Usuario y Arquitectura del Software son de importante conocimiento para que el equipo pueda aplicar satisfactoriamente las pruebas.

2.2.4.1.2. Definir y analizar casos de uso a probar.

Los casos de uso a probar serían las funcionalidades críticas de la aplicación, después de haber hecho un estudio de todos sus requisitos, las mismas son:

- ✓ Gestionar uso de la barrena.
- ✓ Generar cronograma de perforación.
- ✓ Generar Reporte diario de perforación.
- ✓ Gestionar Costo diario (Listar Modificar).

2.2.4.1.3. Diseño del entorno de pruebas.

Identificar y diseñar correctamente el entorno necesario para las pruebas es un paso previo a la realización de las mismas. El entorno de pruebas no es más que las herramientas y máquinas necesarias con el fin de medir la eficiencia de la aplicación a través de las pruebas de carga y estrés.

Las pruebas de software necesitan un entorno particular aislado diferente del entorno de producción, aunque siendo desigual ha de tener las mismas condiciones que éste para así recrear perfectamente la situación con la que ha trabajado el ingeniero y con la que se encontrará el usuario final. Para su identificación se resumen las características más importantes de hardware y software que intervienen en el diseño y aplicación de las pruebas.

CAPÍTULO 2: PROCEDIMIENTO

Software	**Sistema Operativo**	Linux/Debian.
	Herramientas de prueba	OpenLoad.
		Hammerhead.
Hardware	**Memoria RAM**	1GB.
	Microprocesador	Intel(R) Core(TM) 2 Duo CPU 1.80 GHz.
	Disco Duro	80 GB.

Tabla 5 Entorno de pruebas.

2.2.4.2. Diseño.

Después de planificadas las pruebas de carga y estrés; donde se confecciona el Plan de Pruebas, en el cual se definen todos los aspectos relacionados con las pruebas como: Parámetros empleados, Requerimientos de las pruebas, Estrategia de prueba, Recursos, Actividades de prueba, Resultados de las pruebas, Tareas de la etapa de prueba, se pasa a la segunda fase del proceso de prueba; especificando las condiciones para construir dichas pruebas:

- ✓ Diseño de los casos de prueba.
- ✓ Adecuación de la configuración de las herramientas.
- ✓ Determinación de factores de riesgos.

CAPÍTULO 2: PROCEDIMIENTO

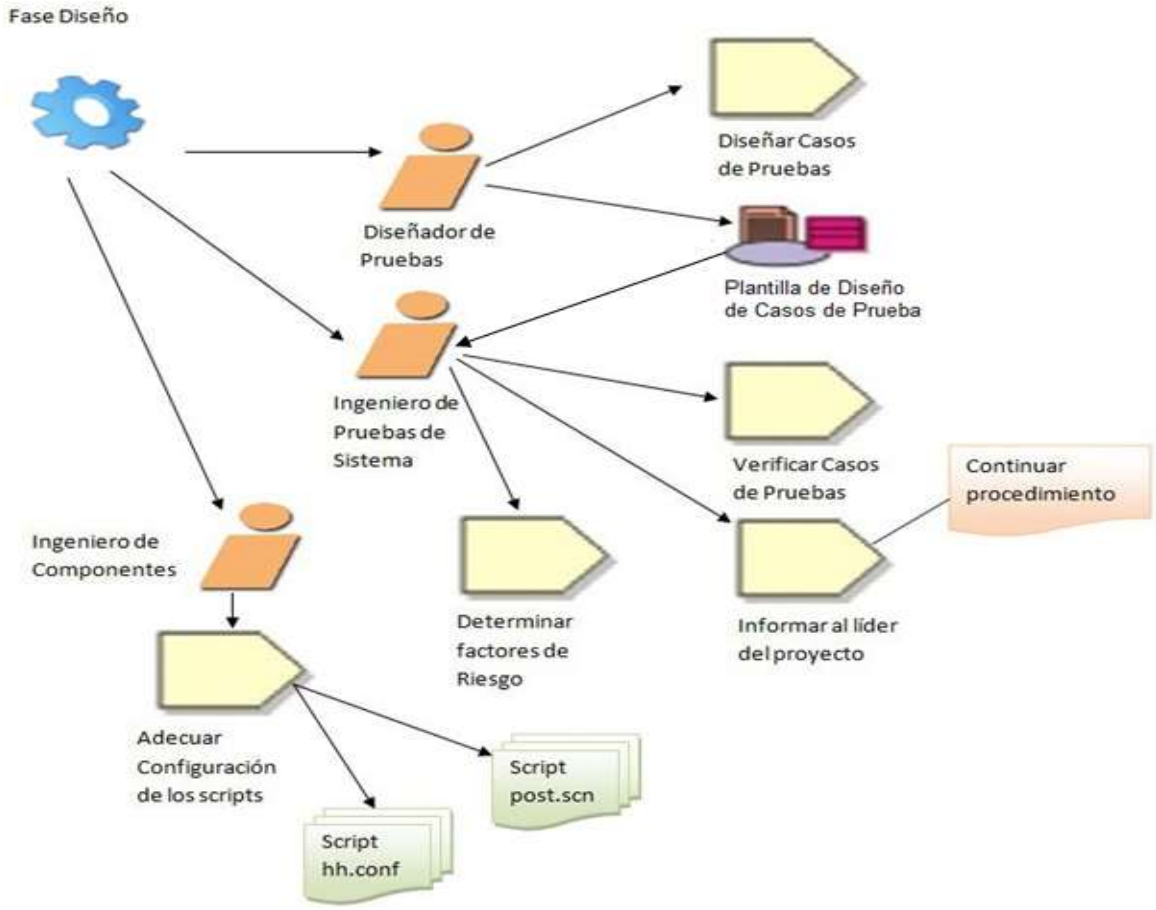


Figura 5 Actividades por roles Fase de Diseño.

2.2.4.2.1. Diseño de los casos de prueba.

El diseño de los casos de prueba es una de las tareas más importantes antes de la realización de las pruebas de estrés y de condiciones de carga. Un diseño erróneo puede llevar a tomar conclusiones equivocadas, positiva o negativamente, respecto a los tiempos de respuestas del sistema.

Los casos de pruebas identificados para pruebas de carga son:

- ✓ CP1_Mostrar uso de la barrena_Carga
- ✓ CP2_Mostrar cronograma de perforación_Carga.
- ✓ CP3_ Reporte diario de perforación_Carga.
- ✓ CP4_Costo diario (Listar Modificar) _Carga.

CAPÍTULO 2: PROCEDIMIENTO

Los casos de pruebas identificados para pruebas de estrés son:

- ✓ CP1_Mostrar uso de la barrena_ Estrés
- ✓ CP2_Mostrar cronograma de perforación_ Estrés.
- ✓ CP3_ Reporte diario de perforación_ Estrés.
- ✓ CP4_Costo diario (Listar Modificar) _ Estrés. (Ver documento *Diseño de Casos de Prueba*)

2.2.4.2.2. Adecuación de la configuración de las herramientas.

Para OpenLoad:

OpenLoad puede ser usado para probar cualquier tecnología de la web del lado del cliente, como Ajax, JavaScript, Applets Java, controles ActiveX, Win Forms, WebForms, Flash entre otras, a ella se puede acceder desde cualquier plataforma que soporte Internet Explorer o Firefox. Permite realizar pruebas de carga desde un solo usuario hasta la simulación de cientos de ellos, este número puede variar según el tipo de aplicación que se esté probando. Mide el rendimiento de la aplicación en tiempo real bajo una carga determinada, lo que permite ver el impacto de los cambios casi de inmediato.

Para realizar pruebas de carga utilizando OpenLoad, lo primero es caracterizar un ciclo de trabajo muy corto de apenas 5 segundos que es el valor que trae la herramienta por defecto y luego de 10 segundos. Posteriormente se especifica la dirección URL de la aplicación, el número de usuarios conectados simultáneamente y se verifica que los tiempos de respuesta se mantengan estables al menos durante los primeros segundos de la prueba. Luego se va aumentando la cantidad de usuarios, hasta obtener el número con el que la aplicación no resiste. (Ver documento *Manual de la herramienta*)

CAPÍTULO 2: PROCEDIMIENTO

Para Hammerhead:

Hammerhead es una herramienta de prueba de estrés diseñada para brindar facilidades como la precisión para determinar la carga máxima que soporta la aplicación. Cuando se ha probado un sitio con Hammerhead y es sometido a prueba nuevamente, la herramienta carga más rápido porque tiene esa dirección guardada en memoria. Los resultados que arroja son los más cercanos posibles a la realidad. Permite probar aplicaciones que empleen como navegador las versiones 2 y 3 de Firefox, mostrando el tiempo de carga de las páginas, ayuda el manejo de la caché, a la iteración con varias direcciones URL y a la recopilación y presentación de los resultados.

Para ejecutar las pruebas de estrés en la herramienta Hammerhead se deben modificar dos ficheros especificando en el de configuración con extensión (conf) datos como: el servidor o IP y el puerto, mientras que en el fichero de escenario con extensión (scn) se tiene en cuenta el protocolo que se estará utilizando y el método que se empleará para la petición HTTP. (Ver documento *Manual de la herramienta*)

2.2.4.2.3. Determinación de factores de riesgos.

Para una realización exitosa de las pruebas, es necesario tomar algunas precauciones como son:

✓ Verificar que las herramientas necesarias para hacer las pruebas se encuentren instaladas correctamente.

✓ Verificar que se va a probar la última versión de la aplicación.

✓ Dar a conocer a los encargados de hacer estas pruebas algunas políticas de salvas para asegurar la información generada.

Teniendo en cuenta la especificación de las variables y los resultados que se van a generar se procede al diseño de las pruebas.

2.2.4.3. Implementación de componentes.

✓ Analizar componentes ficheros.

CAPÍTULO 2: PROCEDIMIENTO

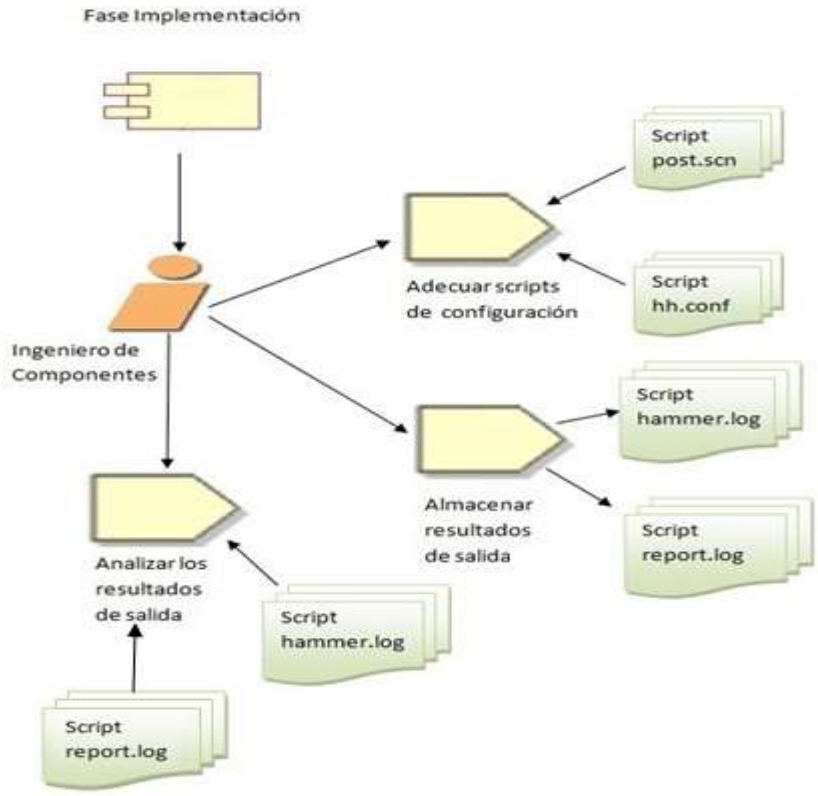


Figura 6 Actividades por roles Fase Implementación de componentes.

2.2.4.3.1 Analizar componentes ficheros.

El resultado final de la implementación es el modelo de implementación, el cual incluye componentes ficheros. Para la realización de las pruebas de estrés utilizando la herramienta Hammerhead se configuran dos ficheros (hh.conf) y (post.scn) necesarios para ejecutar las pruebas y posteriormente a la ejecución de las mismas se generan dos ficheros: hammer.log que en cada línea de registro contiene: sello de tiempo, id de la sesión, nombre del escenario utilizado, tamaño de la respuesta, código de retorno del servidor y la descripción del servidor y report.log donde se registran parámetros como: tiempo de ejecución total, fallas, total de solicitudes atendidas, total de respuestas, promedio de tiempo de respuesta y tiempo de ejecución entre otros.(Ver documento *Manual de la herramienta*)

2.2.4.4. Ejecución.

Se pueden ejecutar las pruebas una vez que esté todo listo, bien definido y el personal esté capacitado para proceder a ejercer las siguientes actividades:

- ✓ Ejecución de las pruebas.
- ✓ Analizar resultados.

CAPÍTULO 2: PROCEDIMIENTO

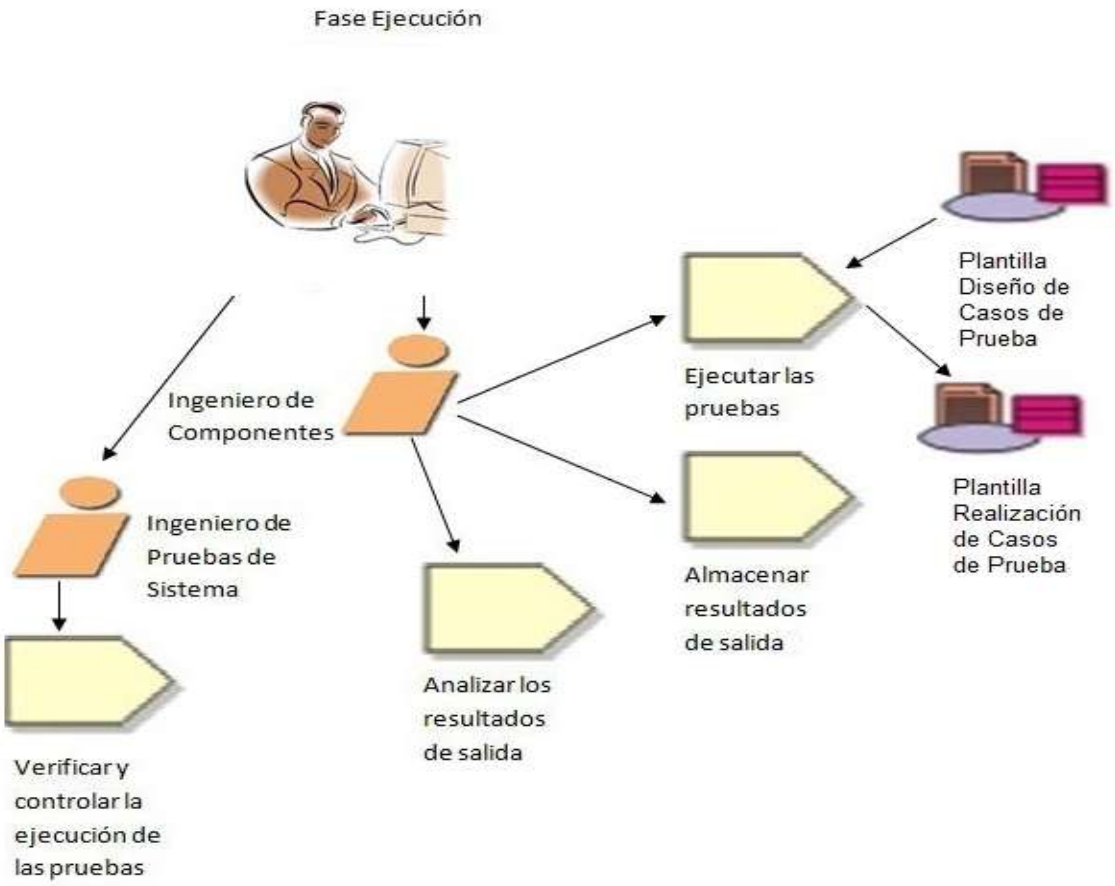


Figura 7 Actividades por roles Fase Ejecución.

2.2.4.4.1. Ejecución de las pruebas.

Se pueden comenzar a ejecutar las pruebas a la aplicación, cuando estén definidos los siguientes aspectos:

- ✓ Configuración del ambiente de pruebas.
- ✓ Instalación de las herramientas automáticas de pruebas.
- ✓ Selección del equipo de probadores.
- ✓ Capacitación del personal involucrado en las pruebas.

CAPÍTULO 2: PROCEDIMIENTO

El personal encargado de hacer las pruebas para comenzar a proceder en la actividad debe tener conocimiento de:

- ✓ Las funcionalidades de la aplicación a probar.
- ✓ Las herramientas que se van a emplear para hacer las pruebas.

Las pruebas comenzarán una vez que se tenga un ambiente listo, bien definido y estén bien diseñados los casos de prueba, con el objetivo de que no aparezcan problemas como son:

- ✓ Demoras por falta de conocimiento de la herramienta.
- ✓ Pruebas mal hechas por falta de un buen diseño de los casos de prueba.
- ✓ Atraso en la implementación de la aplicación a probar o en la entrega de la misma.

Para ejecutar ambas pruebas se deben tener en cuenta aspectos como:

Variables

- ✓ Dirección URL a probar: Variable que indica la dirección de la aplicación a probar.
- ✓ Nivel de concurrencia o número de usuarios: Variable que establece la cantidad de usuarios conectados simultáneamente que simulará la herramienta para la de prueba de carga.
- ✓ Sesiones: número de sesiones simuladas.

2.2.4.4.2. Analizar resultados.

Para realizar la validación de los resultados de las pruebas se verifica la influencia que ejercen algunos parámetros sobre el tiempo de respuesta de la aplicación, elementos que pueden provocar cierto retraso en la generación de la carga en las pruebas e interfieran en la obtención de los resultados.

Ya en la fase final de este procedimiento, se van a analizar los resultados en función de factores como:

Para las pruebas de carga empleando OpenLoad:

- ✓ **Error**: porcentaje de las respuestas con errores.

CAPÍTULO 2: PROCEDIMIENTO

- ✓ **TPS**: transacciones por segundo.
- ✓ **Total Requests**: total de respuestas.

Para las pruebas de estrés utilizando Hammerhead:

- ✓ **Failures:** número de fallos.
- ✓ **NoVerify:** número de verificaciones
- ✓ **Total Requests Served:** total de respuestas del servidor.

Métricas establecidas

- ✓ **Comportamiento aceptado de la CPU:** menor de 30%.
- ✓ **Comportamiento aceptado de la memoria:** menor de 20 MB.
- ✓ **Error aceptado:** 0 – 2 %.

Luego se registran dichos análisis y los resultados obtenidos, en un documento donde se especifican los valores de cada una de las variables que se obtuvieron, se comparan y se organiza la información en el informe de los resultados. (Ver documento *Realización de Casos de Pruebas*)

2.2.4.5. Cierre.

- ✓ Evaluar resultados.
- ✓ Documentar resultados.

CAPÍTULO 2: PROCEDIMIENTO

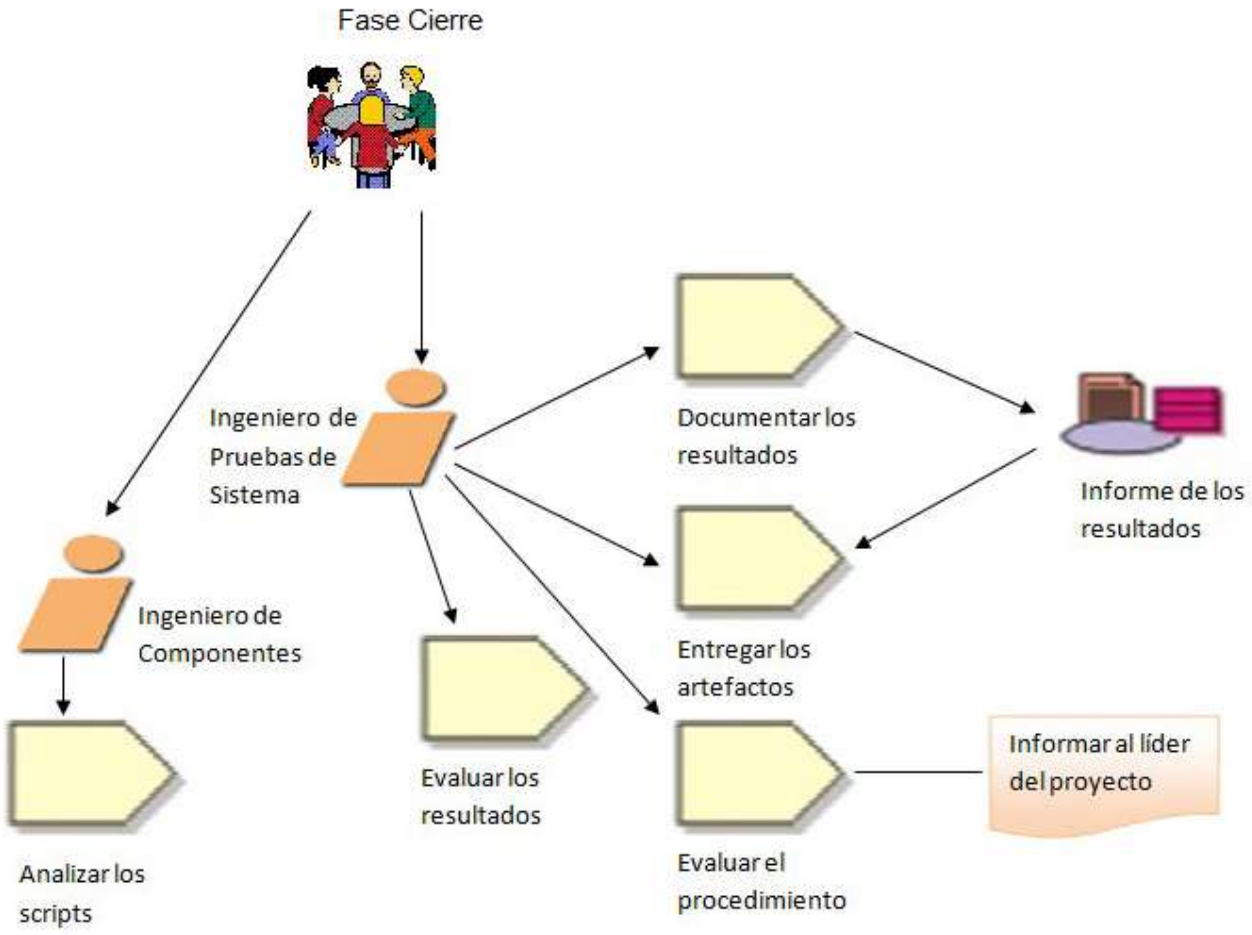


Figura 8 Actividades por roles Fase Cierre.

2.2.4.5.1 Evaluar resultados.

Para determinar el rendimiento del sistema se realizaron pruebas de carga y estrés a los casos de uso más significativos, durante un período de tiempo de 120 segundos. La aplicación se sobrecargó con el número máximo de usuarios conectados simultáneamente, especificados en los requisitos no funcionales.

Con el objetivo de conocer el comportamiento de la aplicación con un número de usuarios conectados concurrentemente, superior a la cantidad esperada se sometió el sistema a pruebas de estrés.

CAPÍTULO 2: PROCEDIMIENTO

Pruebas de Carga.

La ejecución del CP1_Mostrar uso de la barrena_Carga arrojó como resultado que el sistema soporta los 160 usuarios conectados concurrentemente, aunque se observó una disminución considerable del número de transacciones respondidas por segundo (TPS).

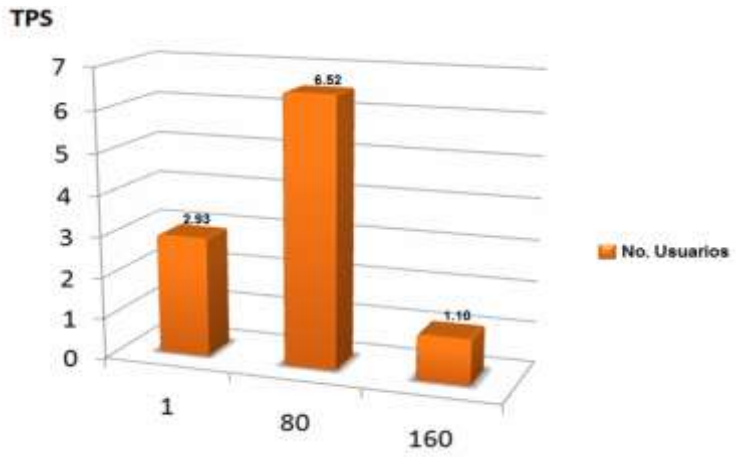


Figura 9 CP1_Mostrar uso de la barrena_Carga.

Durante la ejecución de los casos de prueba CP2_Mostrar cronograma de perforación_Carga (Figura 10) y CP3_ Reporte diario de perforación_Carga (Figura 11) el TPS mantuvo un comportamiento estable para los valores de 80 y 160 usuarios haciendo peticiones al sistema. En el caso de prueba para 1 usuario, el valor TPS fue bajo, ya que el número de peticiones ejecutadas fue mínimo.

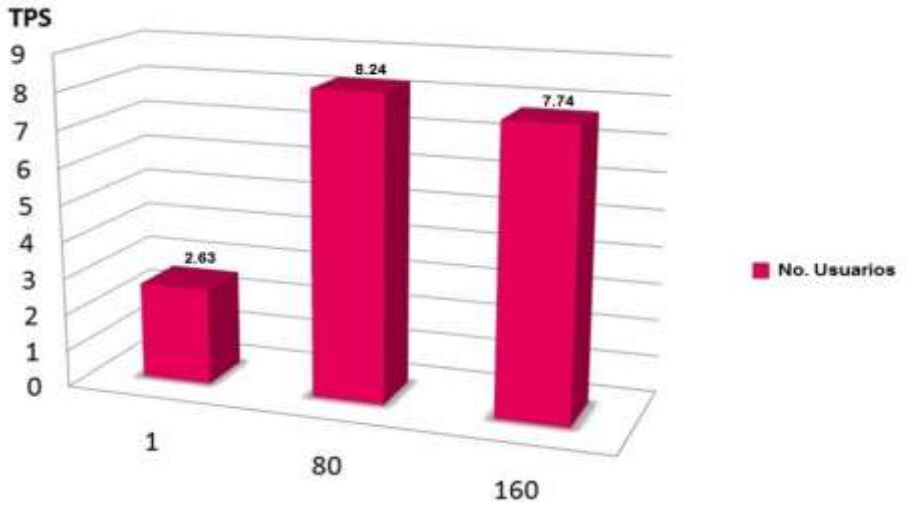


Figura 10 CP2_Mostrar cronograma de perforación_Carga.

CAPÍTULO 2: PROCEDIMIENTO

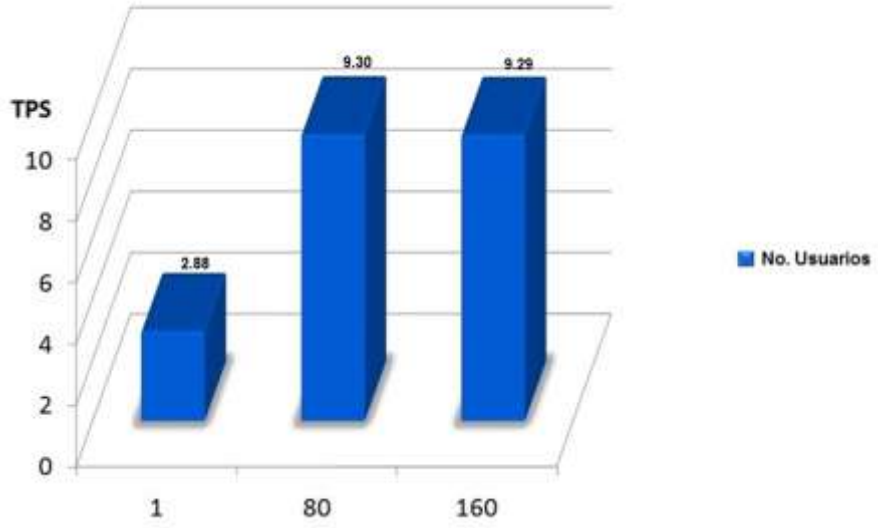


Figura 11 CP3_ Reporte diario de perforación_Carga.

En el caso de prueba CP4_Costo diario (Listar Modificar) _Carga el rendimiento del sistema estuvo bajo para la cantidad de 160 usuarios, lo que evidenció que aunque soporta esta carga, la respuesta no fue eficiente.

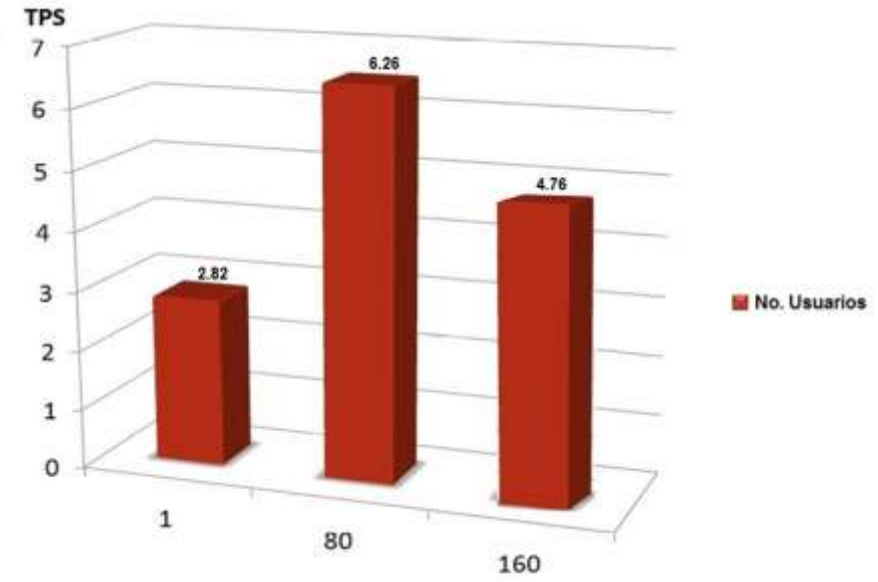


Figura 12 CP4_Costo diario (Listar Modificar) _Carga.

Como resumen de las pruebas de carga, se puede concluir que la aplicación soporta los 160 usuarios conectados simultáneamente. Los porcientos de errores entre transacciones

completadas y transacciones erróneas cumplieron con la métrica establecida en el diseño de los casos de prueba.

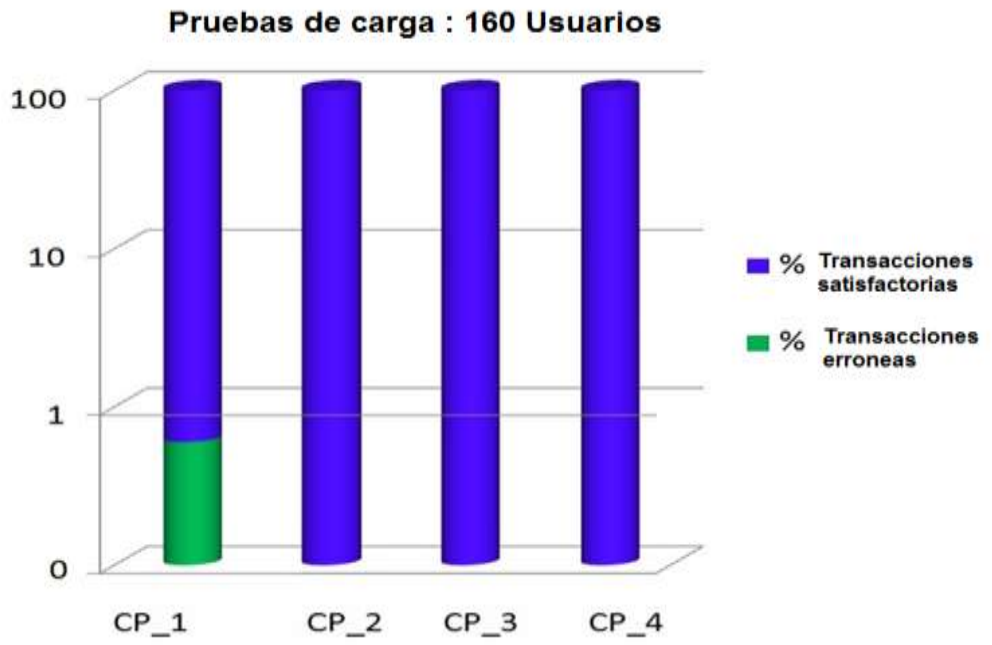


Figura 13 Resumen de Pruebas de Carga.

Pruebas de Estrés.

En la ejecución del caso de prueba CP1_Mostrar uso de la barrena_Estrés, se observó que para 161 y 165 usuarios, el sistema se comportó de manera aceptable, mientras que para 170 usuarios el porciento de respuestas incorrectas superó notablemente la métrica establecida.

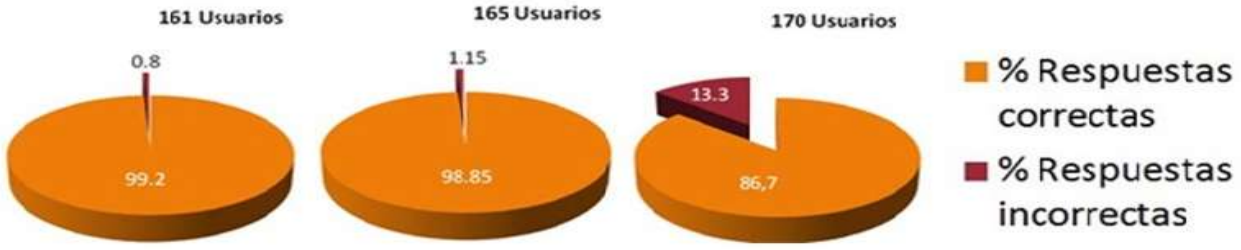


Figura 14 CP1_Mostrar uso de la barrena_Estrés.

Como resultado del caso de prueba CP2_Mostrar cronograma de perforación_ Estrés se obtuvo que para las cantidades de 161, 165 y 170 usuarios el sistema respondió correctamente, con valores dentro de la métrica definida, mientras que para 175 usuarios el error fue del 100%, mostrando que este es el número con el cual colapsa la aplicación.

CAPÍTULO 2: PROCEDIMIENTO

Figura 15 CP2_Mostrar cronograma de perforación_ Estrés.

Durante la ejecución del caso de prueba CP3_ Reporte diario de perforación_ Estrés, se observó que para 180 usuarios la aplicación falló al 100%.Para los casos en que fue sometida a 161, 165, 170 y 175 el porcentaje de error es aceptado dentro del rango establecido.

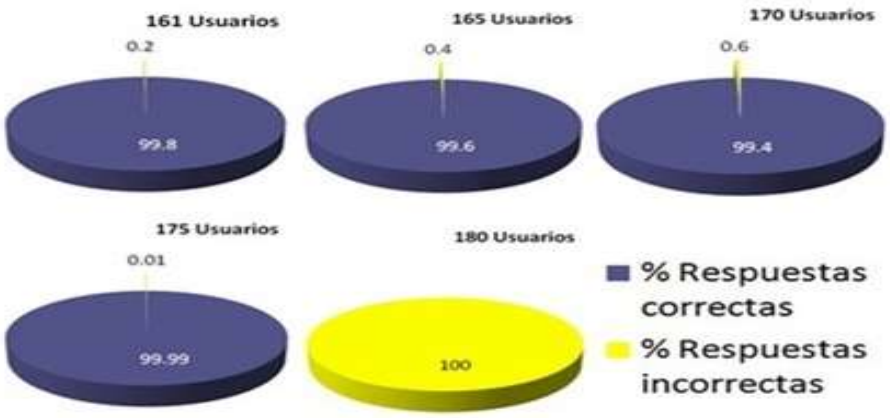


Figura 16 CP3_ Reporte diario de perforación_ Estrés.

Una vez ejecutado el caso de prueba CP4_Costo diario (Listar Modificar) _ Estrés para la cantidad de 161 usuarios el sistema mantuvo un comportamiento satisfactorio, respondió correctamente al 100% todas las peticiones concurrentes realizadas. Bajo una carga de 165 usuarios conectados simultáneamente el número de errores superó la métrica establecida.

CAPÍTULO 2: PROCEDIMIENTO

Figura 17 CP4_Costo diario (Listar Modificar) _ Estrés.

En el resumen de las pruebas de estrés (Figura 18) se puede apreciar que el máximo número de sesiones simultáneas que comprometen la robustez de la aplicación es variable pues para cada caso de prueba el número de usuarios que colapsa la aplicación es distinto .Durante la ejecución de las mismas se observó que el consumo de la CPU del sistema no se afectó, comportándose estable y en el mínimo de valor definido en la métrica. El uso de la memoria física estuvo en un rango entre 5 y 10 MB, por lo que se considera aceptable el uso de los recursos de hardware.

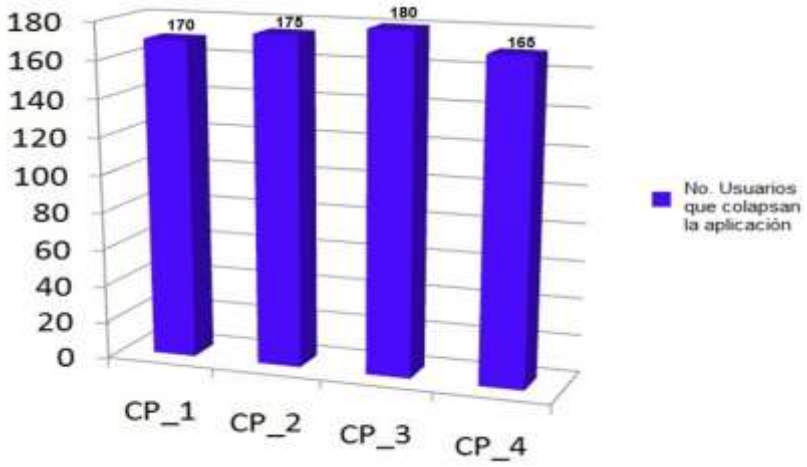


Figura 18 Resumen de las Pruebas de Estrés.

2.2.4.5.2 Documentar resultados.

Los resultados de las pruebas tanto de carga como de estrés serán documentados con el objetivo de evidenciar cuan eficiente o no es el sistema probado y encaminar los esfuerzos hacia la mejora continua de su rendimiento.

CAPÍTULO 2: PROCEDIMIENTO

Conclusiones Parciales.

Durante el presente capítulo se desarrolló detenidamente la propuesta de procedimiento para pruebas de carga y estrés en aplicaciones web, poniéndolo en práctica en el proyecto del SIPP, mostrando detalladamente los pasos a seguir para efectuar dichas pruebas durante la estrategia diseñada. Como parte de la misma se seleccionó el equipo que interviene en la realización de las pruebas, así como una completa descripción de las fases a desarrollar para la correcta ejecución de las pruebas de rendimiento. En cada una de las etapas por las que transcurre el proceso se realizaron actividades como: identificación del entorno de prueba y adecuación de la configuración de las herramientas, teniendo como salida artefactos tangibles que se adjuntan a la investigación como el Plan de Pruebas, Diseño de los casos de pruebas y el Manual de las herramientas OpenLoad y Hammerhead permitiendo así con la descripción y realización de dicho procedimiento validarlo mediante la evaluación de expertos en el tema en el Capítulo 3.

CAPÍTULO 3: VALIDACIÓN DEL PROCEDIMIENTO

CAPITULO 3: VALIDACIÓN DEL PROCEDIMIENTO.

3.1 Introducción.

En el presente capítulo se evaluará el procedimiento para pruebas de carga y estrés en aplicaciones web mediante la selección del tipo de evaluación más adecuado. Se elegirán un grupo de expertos, teniendo en cuenta la experiencia en estos tipos de procesos, a los cuales se les realizará una encuesta para validar la propuesta de solución.

3.2 Tipos de evaluación.

Método de consulta a expertos. Método Delphi: En el año 1963 se diseñó este método que utiliza un grupo de expertos que se mantienen aislados pudiendo ser empleados de la organización o expertos externos. Delphi procede por medio de la interrogación de expertos con el soporte de cuestionarios sucesivos, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos, haciendo hincapié en qué cambios se deben esperar y en qué tiempo (25).

Validación práctica: Consiste en la obtención, comparación y análisis de resultados obtenidos al aplicar prácticamente el procedimiento en varios proyectos.

Recopilación de información: Se basa en recoger estados de opinión, encuestas, cuestionarios o entrevistas a los clientes o a las personas que tengan que ver de una forma u otra con el procedimiento, o con la puesta en práctica de este de forma general.

Grupo focal: Básicamente es la selección de un grupo de personas con conocimientos sobre el tema. Deben ser expertos, de distintos niveles y categorías. Estos se reúnen en un lugar a una hora determinada, donde se discute en forma de grupo debate dirigido por los autores, lo que se quiere conocer sobre el procedimiento (26).

3.3 Selección del método.

Por lo detallado anteriormente se procede a aplicar como tipo de evaluación el método Delphi pues entre sus ventajas reporta:

- ✓ La formación de un criterio con mayor grado de objetividad.
- ✓ El consenso logrado sobre la base de los criterios es muy confiable.
- ✓ La tarea de decisiones, sobre la base de los criterios de expertos, obtenido por este tiene altas probabilidades de ser eficiente.
- ✓ Permite valorar alternativas de decisión.

- ✓ Evita conflictos entre expertos al ser anónimo, (lo que constituye un requisito imprescindible para garantizar el éxito del método) y crea un clima favorable a la creatividad.
- ✓ El experto se siente involucrado plenamente en la solución del problema y facilita su implantación. De ello es importante el principio de voluntariedad del experto en participar en la investigación.
- ✓ Garantiza libertad de opiniones (por ser anónimo y confidencial). Ningún experto debe conocer que a otro se le están solicitando opiniones (27).

3.3.1 Características del Método Delphi.

Anonimato: No debe existir contacto entre los participantes, pero el administrador/gestor de la encuesta puede identificar a cada participante y sus respuestas.

Iteración: Se pueden manejar tantas rondas como sean necesarias.

Retroalimentación controlada: Los resultados totales de la ronda previa no son entregados a los participantes, solo una parte seleccionada de la información circula.

Resultados estadísticos: La respuesta del grupo puede ser presentada estadísticamente (promedios y grado de dispersión)

Figura 19 Proceso Delphi.

CAPÍTULO 3: VALIDACIÓN DEL PROCEDIMIENTO

3.4 Encuesta.

Es una técnica de recogida de información donde, por medio de preguntas escritas organizadas en un formulario impreso, se obtienen respuestas que reflejan los conocimientos, opiniones, intereses, necesidades, actitudes o intenciones de un grupo más o menos amplio de personas; se emplea para investigar masivamente determinados hechos o fenómenos, para conocer opiniones de la población o de colectivos, ya que en su acepción más generalizada, la encuesta implica la idea de la indagación de grupos de individuos y no de sujetos aislados. Lo que interesa es conocer la situación general y no los casos particulares.

Es esencial tener en cuenta que el objetivo de la encuesta es buscar información a través de preguntas directas e indirectas, las cuales se organizan con determinados requisitos metodológicos en un cuestionario. O sea, la encuesta es la técnica y el cuestionario es el instrumento a través del cual se interroga a la población.

Si la encuesta está encaminada a obtener información pertinente y significativa para una investigación, debe ser elaborada atendiendo al diseño investigativo, en correspondencia con el problema, el objetivo, la hipótesis (en caso de que esta sea formulada), las categorías, variables e indicadores definidos y el marco teórico que sustenta dicho diseño. Al planear la encuesta puede resultar de utilidad atender a los siguientes aspectos, como guía metodológica orientadora y flexible que contempla una serie de tareas lógicamente concatenadas, pero que en ningún caso debe ser empleada como un procedimiento algorítmico:

1. Definir el objetivo de la encuesta. Se debe tener claro que información se requiere.
2. Definir las variables que son sujeto de investigación de acuerdo con el objetivo de la encuesta.
3. Definir los indicadores ya que para obtener la información de las variables, es necesario realizar varias mediciones.
4. Construir el cuestionario. Para construir un cuestionario, es importante que las preguntas se relacionen exclusivamente con las variables y los indicadores definidos previamente.
5. Seleccionar la población objetiva.
6. Realizar la encuesta.
7. Recopilar la información de los cuestionarios y representarlas en tablas y gráficos.
8. Obtener las conclusiones del estudio de acuerdo con los objetivos planteados (28). (Ver Anexo 2).

CAPÍTULO 3: VALIDACIÓN DEL PROCEDIMIENTO

3.5 Selección del grupo de expertos.

Se seleccionaron 7 expertos, tomando como criterio de selección la efectividad de la actividad profesional que realizan, ser ingeniero o de un grado superior, la experiencia que poseen en el desarrollo de pruebas de software sobre aplicaciones web y los años vinculados a esta actividad.

3.6 Coeficiente de Concordancia de Kendall y prueba de hipótesis.

El coeficiente de Kendall es una de las técnicas no paramétricas para medir el grado de correlación entre las variables de una muestra. Este coeficiente mide el grado de asociación entre varios conjuntos (k) de N entidades. Y es útil para determinar el grado de acuerdo entre varios jueces, o la asociación entre tres o más variables. Este método de pronóstico es importante porque brinda un modelo para la ordenación de entidades de acuerdo a un consenso, cuando no hay un orden objetivo de las entidades.

En la prueba estadística Coeficiente de Concordancia de Kendall (W), ofrece el valor que posibilita decidir el nivel de concordancia entre los expertos. El valor de W oscila entre 0 y 1. El valor de 1 significa una concordancia de acuerdos total y el valor de 0 un desacuerdo total. La tendencia a 1 es lo deseado pudiéndose realizar nuevas rondas si en la primera no es alcanzada significación en la concordancia (29).

Este coeficiente se calcula con el uso de las fórmulas que muestra la figura siguiente:

$$W = \frac{S}{\frac{1}{12}k^2\left(N^3 - N\right) - k\sum_T T} \qquad S = \sum_{j=1}^{N}\left(R_j - \frac{\sum_{j=1}^{N} R_j}{N}\right)^2 \qquad T = \frac{\sum\left(t^3 - t\right)}{12}$$

Figura 20 Fórmulas para el cálculo del coeficiente de Kendall.

Donde:

W: Coeficiente de concordancia.

K: Cantidad de expertos.

N: Cantidad de variables.

T: Factor de corrección.

Rj: Suma de los rangos asignados a cada variable.

S: Suma de los cuadrados de las desviaciones.

t: Número de observaciones en un grupo ligado por un rango dado.

No basta con saber si W está más próximo a 0 o 1 sino que además debemos saber si W es significativamente distinta de 0 para rechazar la hipótesis de concordancia casual. Esta prueba sería en principio una prueba de hipótesis.

Hipótesis:

H0: No hay concordancia (entre los expertos, jueces, muestras, etc.)

H1: Hay concordancia.

Para muestras pequeñas (N ≤ 7) se usa como estadígrafo el numerador del coeficiente de Kendall: S. La distribución de este estadígrafo se ha obtenido y se han tabulado ciertos valores críticos en una tabla denominada R (Ver Anexo 3). Dicha tabla contiene valores de S para la significación de W en los niveles 0.05 y 0.01. Y se rechazará la hipótesis nula si el valor de S calculado es mayor que el obtenido en la tabla.

3.6.1 Análisis de la concordancia.

Con el objetivo de conocer el nivel de aceptación que tiene el procedimiento confeccionado, así como el nivel de acuerdo entre cada uno de los expertos, se analizan en este epígrafe los resultados obtenidos. Para ello se tienen en cuenta el cálculo del Coeficiente de Concordancia de Kendall y la prueba de hipótesis (explicados anteriormente), a partir de la evaluación que dio cada especialista al procedimiento en el segundo cuestionario.

Hipótesis a tener en cuenta.

H0: No hay concordancia entre los expertos.

H1: Hay concordancia entre los expertos.

CAPÍTULO 3: VALIDACIÓN DEL PROCEDIMIENTO

Expertos\Criterios	Criterio # 1	Criterio # 2	Criterio # 3	Criterio # 4	Criterio # 5
Experto 1	5	5	4	4	5
Experto 2	5	5	4	5	5
Experto 3	4	4	5	5	5
Experto 4	5	4	5	5	5
Experto 5	5	3	4	5	5
Experto 6	5	4	5	5	5
Experto 7	5	4	5	5	5
Experto 8	5	4	5	5	5
Promedio	4,875	4,125	4,625	4,875	5
Desviación Estándar	0,35355339	0,640869944	0,51754917	0,35355339	0
Promedio general	4,68571429				

Tabla 6 Resultados de las evaluaciones de los expertos.

Expertos\Criterios	Criterio # 1	Criterio # 2	Criterio # 3	Criterio # 4	Criterio # 5
Experto 1	4	4	1,5	1,5	4
Experto 2	3,5	3,5	1	3,5	3,5
Experto 3	1,5	1,5	4	4	4
Experto 4	3,5	1	3,5	3,5	3,5
Experto 5	4	1	2	4	4
Experto 6	3,5	1	3,5	3,5	3,5
Experto 7	3,5	1	3,5	3,5	3,5
Experto 8	3,5	1	3,5	3,5	3,5

Tabla 7 Rangos de puntaje ligados.

Rj	27	14	22,5	27	29,5
Rj-∑Rj/N	3	-12	-1,5	3	5,5
(Rj-∑Rj/N)^2	9	144	2,25	9	30,25

Tabla 8 Cálculo de S.

T	t1	t2	t3	t4	t5	t6	t7	t8
32	2,5	5	2,5	5	2	5	5	5

Tabla 9 Cálculo de T.

Finalmente, el valor de W calculado fue de **0.73** y el valor de S fue de **194,5**. El valor de S obtenido en la tabla R (Ver anexo 3) para un nivel de significación de 0.05 es de **183,7** que comparado con el valor de S calculado, es menor. Esto indica que se rechaza la hipótesis nula,

lo que finalmente revela que hay concordancia entre los especialistas. Siendo aceptado el procedimiento con un promedio de **4,69**.

3.6.2 Análisis de los resultados.

De la validación dada por cada experto se obtuvieron los siguientes resultados estadísticos. Estos indican que hubo un buen promedio de evaluación y con ello de aceptación del procedimiento.

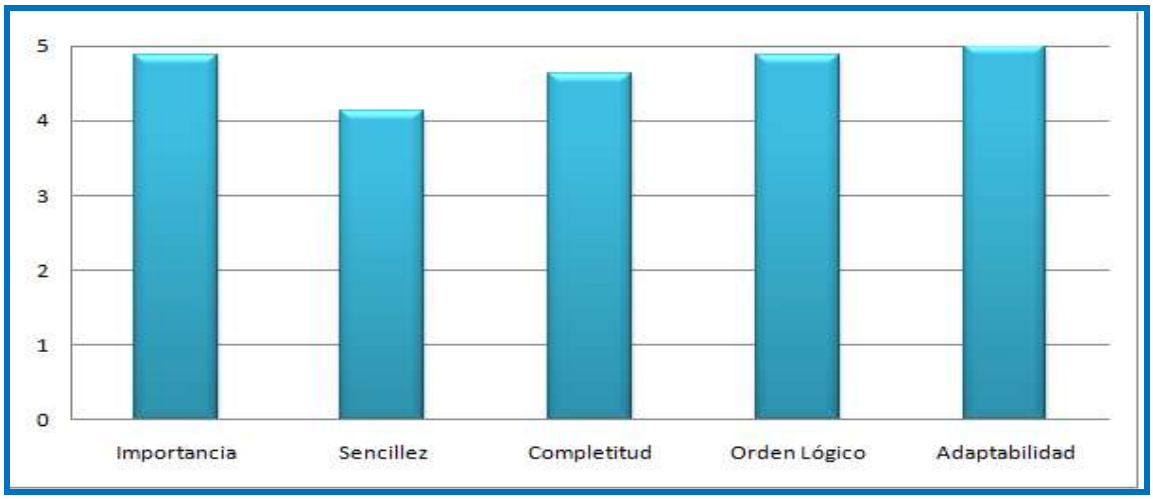


Figura 21 Promedio por criterios de evaluación.

El Grado de unicidad de las evaluaciones por criterios (Figura 22) da una medida del nivel de desacuerdo que tuvieron los expertos para evaluar el procedimiento. Se destaca el criterio de adaptabilidad, en el que todos los expertos tuvieron un acuerdo total. Mientras los restantes criterios oscilaron entre 0.3, 0.5 y 0.6.

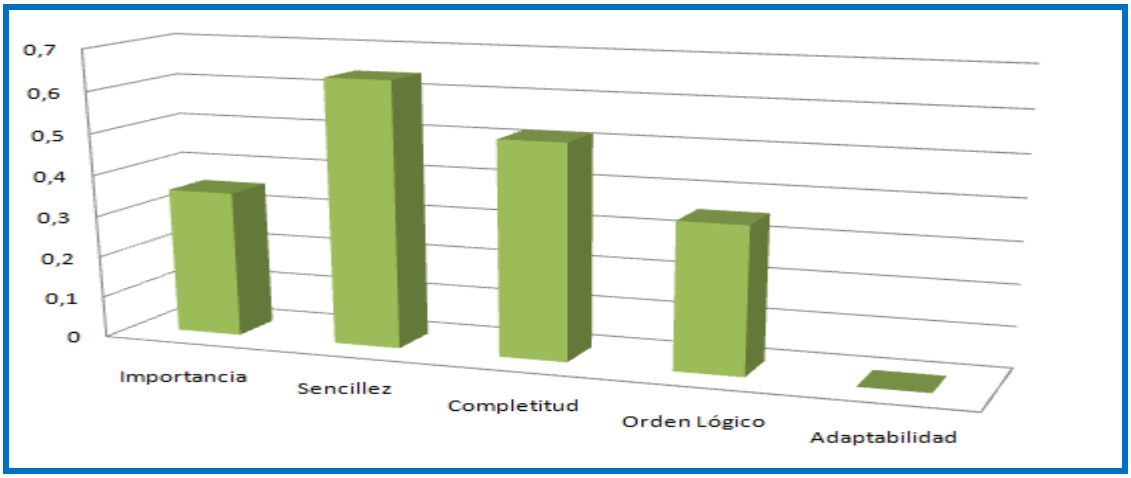


Figura 22 Grado de unicidad de las evaluaciones por criterios.

CAPÍTULO 3: VALIDACIÓN DEL PROCEDIMIENTO

Conclusiones Parciales.

En este capítulo mediante la selección de un grupo de expertos con conocimiento en el tema que se investiga se validó el procedimiento. La propuesta tuvo buena aceptación, ya que a través del cálculo del Coeficiente de Kendall se comprobó que las opiniones de los expertos coincidían notablemente, destacando el criterio de adaptabilidad de la solución propuesta.

Conclusiones Generales.

Se propuso un procedimiento caracterizado por completitud, sencillez y adaptabilidad, capaz de dar solución al problema planteado en la investigación. Se realizó un profundo estudio del arte centrado en las características de las diversas herramientas existentes a nivel mundial que permiten aplicar pruebas automatizadas de carga y estrés, seleccionando entre ellas a OpenLoad y Hammerhead. Se empleó la metodología RUP para regir el proceso de pruebas en la generación de los artefactos necesarios en cada una de las fases propuestas. La misma se aplicó al proyecto SIPP, arrojando resultados que permiten mejorar el sistema en cuanto a su rendimiento.

La solución descrita fue validada mediante la selección de un panel de expertos con alto conocimiento sobre la temática en cuestión. A través de los resultados obtenidos en los cuestionarios se concluyó que el procedimiento posee un alto valor de importancia. Con la aplicación del cálculo de Kendall y de la prueba de hipótesis se observó que el nivel de concordancia entre los expertos fue significativo, quedando demostrada la validez del procedimiento.

Recomendaciones

- ✓ Continuar investigando sobre otras herramientas disponibles para Linux como sistema operativo, que muestren el uso de los recursos de hardware durante la ejecución de las pruebas en aplicaciones web.
- ✓ Aplicar en los proyectos productivos de la Universidad las pruebas de carga y estrés utilizando las herramientas OpenLoad y Hammerhead, con la ayuda de un previo estudio realizado a los manuales de las herramientas.
- ✓ Dar seguimiento al procedimiento propuesto con la finalidad de perfeccionarlo.

Referencias Bibliográficas

Referencias Bibliográficas.

1. Solari Martin, Prueba de software. Disponible en: http://athenea.ort.edu.uy/publicaciones/ingsoft/ortsf/areas/IntroduccionPrueba.pdf

2. LLopis Miguel, Serie ECDB: ¿En qué consiste el proceso de pruebas de software?, 9 de febrero de 2008. Disponible en: http://geeks.ms/blogs/mllopis/archive/2008/02/09/191-en-qu-233-consiste-el-proceso-de-pruebas-de-software.aspx

3. Disponible en: http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php

4. Conferencia 7 de Ingeniería de Software ll, Curso 2009-2010.Disponible en: http://eva.uci.cu/file.php/259/Curso_20092010/Conferencia_7/Materiales_basicos/Conferencia_7_Disciplina_Prueba.doc

5. Valencia Eugenia María. Disponible en: http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/PRUEBASoftware.pdf

6. Disponible en: http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf

7. Universidad Distrital Francisco José de Caldas, Grupo de Investigación Arquitecturas de software, Visión general de las pruebas de software. Disponible en: http://www.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=80&type=1

8. Acuña Brito Karenny. Metodologías tradicionales y metodologías ágiles. Disponible en: http://www.eumed.net/libros/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm

9. Jacobson Ivar, Booch Grady, Rumbaugh James, El proceso unificado de desarrollo de software. 2003. Madrid: Pearson Education S.A.

10. Soluciones y servicios .Pruebas de rendimiento de aplicaciones. http://www.serikat.es/pdf/revista_14/revista_14_03.pdf

11. Testhouse. Pruebas de Rendimiento, 2010. Disponible en: http://es.testhouse.net/index.php/servicios/pruebas-no-funcionales/pruebas-de-rendimiento

12. Abel Ignacio, Giampedraglia Pablo, Análisis e implementación de un Toolkit para Testing de Performance, marzo 2005. Disponible en: http://www.fing.edu.uy/inco/pedeciba/bibliote/tgradoinf/tg-abel.pdf

Referencias Bibliográficas

13. Conferencia 7 de Ingeniería de Software ll, Curso 2009-2010.Disponible en : http://eva.uci.cu/file.php/259/Curso_20092010/Conferencia_7/Materiales_basicos/Conferencia_7_Disciplina_Prueba.doc

14. Torossi Gustavo. El Proceso Unificado de Desarrollo de Software. Disponible en:

http://antares.itmorelia.edu.mx/~jcolivar/courses/pm10a/rup.pdf

15. biblioteca personal de la tecnología del Ðð©'§, Ventajas y pauta de la prueba automatizada. Disponible en: http://www.docsdownloads.com/spanish.php?u=tech/33667.php

16. Sucari Ariel. Pruebas ágiles.1 de mayo de 2007.Disponible en: http://www.sg.com.mx/content/view/268

17. vPerformer, 2009. Disponible en: http://www.verisium.com/products/vPerformer/index.html

18. Sitios de descargas de software. WebLoad Professional. Disponible en: http://www.freedownloadmanager.org/es/downloads/WebLOAD_Professional_59025_p/

19. OpenLoad Tester by Open Demand System, 2007.Disponible en: http://www.als-es.com/home.php?location=herramientas/entorno-pruebas/open-load-tester

20. SoftQanetwork. Pruebas de Regresión. Disponible en: http://www.softqanetwork.com/2008/05/pruebas-de-regresion/

21. Sitios de descargas de software. Herramienta de estrés para servidores web (Webserver Stress Tool). Disponible en: http://www.freedownloadmanager.org/es/downloads/Herramientas_de_Tensi%C3%B3n_de_Webserver_10183_p/

22. Web Application Testing WAPT, 2003-2009 .Disponible en: http://www.loadtestingtool.com/download.shtml

23. Hammerhead (1). Disponible en: http://linux.die.net/man/1/hammerhead

24. Expediente del proyecto SIPP.

25. Cabezas Trujillo Raúl. Aplicaciones del Método Delphi. Disponible en:

http://administracion.uexternado.edu.co/posgrado/espep/matdi/GENERAL/prospectiva%20tecnologica/delphiRaulTrujilloCabezas.pdf

26. Hernández Tamayo Juniel, Pardo Miranda Daily. Procedimiento para el despliegue de soluciones web. Universidad de las Ciencias Informáticas. Ciudad Habana, 2009.159

27. Iglesias Moráguez Arabel Ing. El Método Delphi, 2005-2006.Disponible en:

Referencias Bibliográficas

http://www.gestiopolis.com/canales6/eco/metodo-delphi-estadistica-de-investigacion-cientifica.htm

28. Simons Castellanos Beatriz DRA. La encuesta y la entrevista en la investigación educativa. Cuba, 1998. 34

29. Universidad de las Ciencias Informáticas. Conferencia 8: Dócimas no Paramétricas. Bondad de Ajuste. Homogeneidad de la muestra. Análisis para evaluar concordancia, 2010.Disponible en: http://eva.uci.cu/file.php/69/Tema_4/Act.%2022.pdf

Bibliografía

Bibliografía.

Zarzuela Ferrer Jorge. Metodologías ágiles. Disponible en: http://libresoft.dat.escet.urjc.es/html/downloads/ferrer-20030312.pdf

Canós H. José, Letelier Patricio, Penadés Carmen María. Metodologías Ágiles en el Desarrollo de Software. Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.pdf

Acuña Brito Karenny. Metodologías tradicionales y metodologías ágiles. Disponible en:

http://www.eumed.net/libros/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm

Infante Luis. Sobre las metodologías tradicionales de desarrollo de sistemas, 2009.Disponible en: http://www.bi-la.com/profiles/blogs/sobre-las-metodologias

El Proceso Unificado de Desarrollo de Software (RUP). Disponible en: http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm

Justificación del uso del proceso unificado de desarrollo de software como modelo. Disponible en: http://www.intep.edu.co/intep3/f_docente/66887/Proceso%20Unificado.pdf

El Proceso Unificado de Desarrollo de Software. Disponible en: http://docs.google.com/viewer?a=v&q=cache:PMePXkltJjgJ:antares.itmorelia.edu.mx/~jcolivar/courses/pm10a/rup.pdf+el+proceso+unificado+de+desarrollo+de+software

Seguridad de las aplicaciones Web. 1999 - 2009. Disponible en: http://www.sowre.es/wps/portal/?WCM_GLOBAL_CONTEXT=/WebPublica/Servicios/Seguridad+Web/

Optima Technology. Disponible en: http://www.optima.com.mx/webload.htm

Herramientas de pruebas de rendimiento .Disponible en: http://www.opensourcetesting.org/performance.php

Disponible en: http://spanish.downloadsportal.com/webmasters/administracion/wapt_application-24843.html

Sitios de descargas de software. WebLoad Professional. Disponible en: http://www.freedownloadmanager.org/es/downloads/WebLOAD_Professional_59025_p/

RadView. System Requirements. Disponible en: http://www.radview.com/product/System-Requirments.aspx

SGuía, 2009.Disponible en: http://www.sg.com.mx/guia/node/795

Hammerhead FAQ. Disponible en: http://www.stevesouders.com/hammerhead/faq.php

Bibliografía

Sitios de descargas de software. Herramienta de estrés para servidores web (Webserver Stress Tool). Disponible en: http://www.freedownloadmanager.org/es/downloads/Herramientas_de_Tensi%C3%B3n_de_Webserver_10183_p/

Hammerhead. Disponible en:http://webscripts.softpedia.com/script/Web-Hosting-Tools/Hammerhead-28436.html

http://www.promonegocios.net/mercadotecnia/encuestas-tipos.html

http://www.ejemplode.com/11-escritos/123-ejemplo_de_encuesta.html

http://www.foroswebgratis.com/mensaje-re_pasos_para_realizar_una_encuesta-9779-1092787-1-5584987.htm

Anexos

Anexos.

Anexo 1 Encuesta aplicada a los líderes de proyecto.

¿Sabes que son las pruebas de carga? ___Si __No

¿Sabes que son las pruebas de estrés? ___Si __No

¿A tu proyecto se le aplican pruebas de carga? ___Si __No __ No se __Manual ___Automática

¿A tu proyecto se le aplican pruebas de estrés? ___Si __No __ No se __Manual ___Automática

¿Conoces algún proyecto al que se les apliquen? ___Si __No

_____Nombre del Proyecto. __Manual ___Automática

¿Conoces alguna herramienta automatizada para las pruebas de carga?

___Si __No _____Nombre de la herramienta.

¿Conoces alguna herramienta automatizada para las pruebas de estrés?

___Si __No _____Nombre de la herramienta.

Anexo 2 Diseño de las encuestas para determinar el coeficiente de competencia de los expertos.

En la presente tesis, se desea someter a la valoración de un grupo de expertos una propuesta de procedimiento para pruebas de carga y estrés en aplicaciones web, para garantizar la eficiencia del software que se produce en la universidad. Para ello necesitamos conocer el grado de dominio que usted posee sobre el tema y con ese fin se desea que responda lo que se le pide a continuación.

Vinculación a Proyecto: _____

Especialidad _____

Años de experiencia _____

Categoría docente _____

Categoría científica _____

Anexos

1- Marque con una cruz (X) el grado de conocimiento que UD. tiene sobre la temática que se investiga:

1	2	3	4	5	6	7	8	9	10

2.- Marque con una cruz (X) las fuentes que le han servido para argumentar el conocimiento que tiene UD. de la temática que se investiga.

No.	Fuentes de adquisición del conocimiento.	Grado de Influencia		
		Alto	Medio	Bajo
1	De forma autodidacta.			
2	Trabajos de autores nacionales.			
3	Trabajos de autores extranjeros			
4	Eventos científicos acerca del tema.			

Gracias por su ayuda.

Cuestionario # 2.

Este cuestionario se elaboró para considerar la evaluación cuantitativa de cada uno de los expertos, según diferentes criterios, los cuales se citan a continuación:

Criterio #1- Importancia del uso del procedimiento.

Criterio #2 - Sencillez del procedimiento.

Criterio #3 - Completitud del procedimiento.

Criterio #4 - Orden lógico de las etapas, actividades y tareas.

Criterio #5 - Adaptabilidad del procedimiento a las características del producto SIPP.

Anexos

Anexo 3 Tabla R.

Expertos (K)	Características (N)					Valores adicionales para N = 3x	
	3	4	5	6	7	M	S
α = 0.05							
3			64.4	103.9	157.	9	54.0
4		49.5	88.4	143.3	217.0	12	71.9
5		62.6	112.3	182.4	276.2	14	83.8
6		75.7	136.1	221.4	335.2	16	95.8
8	48.1	101.7	183.7	299.0	453.1	18	107.7
10	60.0	127.8	231.2	376.7	571.0		
15	89.8	192.9	349.8	570.5	864.9		
20	119.7	258.0	468.5	764.4	1158.7		
α = 0.01							
3			75.6	122.8	185.6	9	75.9
4		61.4	109.3	176.2	265.0	12	103.5
5		80.5	142.8	229.4	343.8	14	121.9
6		99.5	176.1	282.4	422.6	16	140.2
8	66.8	137.4	242.7	388.3	579.9	18	158.6
10	85.1	175.3	309.1	494.0	737.0		
15	131.0	269.8	4757.2	758.2	1129.5		
20	177.0	364.2	541.2	1022.2	1521.9		

Glosario de Términos

Glosario de Términos.

---A---

ActiveX: Es una tecnología de Microsoft para el desarrollo de páginas dinámicas. Tiene presencia en la programación del lado del servidor y del lado del cliente.

Adobe Flex: Es un término que agrupa una serie de tecnologías publicadas desde marzo de 2004 por Macromedia para dar soporte al despliegue y desarrollo de Aplicaciones Enriquecidas de Internet, basadas en su plataforma propietaria Flash.

Artefacto: Productos tangibles del proyecto que son producidos, modificados y usados. Pueden ser modelos, elementos dentro del modelo, documentos, gráficos, código fuente y ejecutables.

Ajax: Acrónimo de Asynchronous JavaScript And XML es una técnica de desarrollo web para crear aplicaciones interactivas.

Applet Java: Es un programa que puede incrustarse en una página web lo que permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la página web en su navegador.

---B---

BD: Base de datos.

Business Intelligence: Se denomina **inteligencia empresarial**, **inteligencia de negocios** o **BI** (del inglés business intelligence) al conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos existentes en una organización o empresa.

---C---

Calidad**:** La capacidad de cumplir con las características inherentes de un producto, componente de producto, o proceso para satisfacer las exigencias de los clientes.

CALISOFT: Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos

Caso de prueba (CP): Conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular, por ejemplo, ejercitar un camino concreto de un

programa o verificar el cumplimiento de un determinado requisito. También se puede referir a la documentación en la que se describen las entradas, condiciones y salidas de un caso de prueba.

CEDIN: Centro de Desarrollo de Informática Industrial. División Estructural de la Facultad 5 encaminado a organizar mejor la producción en cuanto a los temas a desarrollar.

CICPC: Cuerpo de Investigaciones Científicas, Penales y Criminalística de la República Bolivariana de Venezuela.

Componentes: Parte discreta de un sistema capaz de operar independientemente, pero diseñada, construida y operada como parte integral del sistema.

CPU: Central Processing Unit (Unidad Central de Procesamiento). Es el componente en un ordenador, que interpreta las instrucciones y procesa los datos contenidos en los programas de la computadora.

Cuellos de botella: Esta teoría está basada en que los procesos de cualquier ámbito, se mueven a una velocidad lenta; y la manera de balancear el proceso es utilizar un acelerador para lograr que trabaje hasta el límite de su capacidad para acelerar el proceso completo, estos factores limitantes se denominan restricciones, embudos o cuellos de botella.

CUPET: Unión Cuba petróleo.

---D---

Defectos*:* Consecuencia de un error. Incumplimiento de un requisito asociado a un uso previsto o especificado.

DIPP: Dirección de Intervención y Perforación de Pozos.

---E---

Eficiencia: Medida del grado en que una actividad alcanza sus objetivos, optimizando el uso de los recursos disponibles.

Estrategia**:** Es el conjunto de pasos que indican cómo desarrollar de forma organizada las pruebas a un software o aplicación determinada.

Glosario de Términos

***Estructura*:** Distribución y orden con que está compuesta una obra.

ERP / CRM Applications: CRM Integrado de aplicaciones ERP. Se centra en integrar de manera eficiente los procesos de negocio, utilizando una base de datos segura y centralizada.

---F---

Falla: Error en el funcionamiento que emite el sistema.

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

FreeBSD: Es un avanzado sistema operativo, desarrollado y mantenido por un numeroso equipo de personas; derivado de BSD la versión de UNIX® realizada en la Universidad de California, Berkeley.

---G---

GB: Giga byte.

---H---

Hardware: Conjunto de componentes físicos de una computadora. Refiérase a objetos tangibles y palpables como son los discos, lectores de discos, monitores, teclados, las impresoras, tarjetas y chips.

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

HTTPs: Protocolo de seguridad para navegadores.

---L---

Linux: Es un software libre; todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General de GNU) y otra serie de licencias libres.

Glosario de Términos

---M---

MB: Mega byte.

Metodología: Es un conjunto de procedimientos, técnicas, instrumentos y documentos que ayudan a los analistas y programadores en sus esfuerzos para obtener un nuevo sistema informativo. Consiste en fases que guían al diseñador en la elección de las técnicas más apropiadas en cada paso del proyecto y lo ayudan a planificar, dirigir, controlar y evaluar el mismo.

Métricas: Proporcionan información objetiva que contribuye al mejoramiento de los procesos y productos de software.

---P---

***Plan de Pruebas*:** Es una guía que describe las estrategias, los recursos y la planificación de las pruebas, es el principal factor del éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel.

Procedimiento: Forma específica de llevar a cabo una actividad. En muchos casos los procedimientos se expresan en documentos que contienen el objeto y el campo de aplicación de una actividad; que debe hacerse y quien debe hacerlo; cuando, donde y como se debe llevar a cabo; que materiales, equipos y documentos deben utilizarse; y como debe controlarse y registrarse.

Proceso: Un proceso se define como un conjunto de tareas, actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido

Producto: Es cualquier cosa que puede ser ofrecida al mercado para su compra, para su utilización o para su consideración. Es cualquier bien, servicio o idea capaz de motivar y satisfacer a un comprador.

Proyecto: Elemento organizativo a través del cual se gestiona el desarrollo del software, el resultado de un proyecto es una versión del producto.

Glosario de Términos

Prueba de software: Ejecución de un sistema bajo condiciones específicas, se observan y se analizan los resultados realizándose una evaluación de los mismos.

---R---

***RAM: Random Access Memory (Memoria de Acceso Aleatorio)*.** Es la que se encarga de almacenar la información que el computador está utilizando mientras se encuentra encendido.

Recursos*:* Son todos aquellos elementos necesarios, tanto tangibles como intangibles, para que una organización cumpla con sus objetivos.

Requerimientos: Una condición o capacidad que debe estar presente en el sistema o componentes del sistema para satisfacer un contrato estándar, especificación u otro documento formal.

Roles: Papel que ejerce un actor en una actividad o proyecto.

RUP: Rational Unified Project, (Proceso Unificado de Desarrollo).

---S---

***Scripts*:** Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de órdenes y usualmente son archivos de texto. También un *script* puede considerarse una alteración o acción a una determinada plataforma.

SIPP: Sistema de Información de Perforación de Pozos Petroleros.

Sistema: Conjunto de elementos relacionados que interactúan entre sí para lograr un fin determinado.

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

Solaris: Es un sistema operativo de tipo Unix desarrollado desde 1992 inicialmente por Sun Microsystems y actualmente por Oracle Corporation como sucesor de SunOS.

---T---

Glosario de Términos

TPS: Transacciones por segundo.

---U---

UCI*:* Universidad de las Ciencias Informáticas.

UD: Usted.

***UML: Unified Modeling Language (Lenguaje Unificado de Modelado*).** Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.
URL: Localizador de Recurso Uniforme. Dirección global de documentos y de otros recursos.

***Usuario*:** Humano que interactúa con un sistema.

---V---

***Validación*:** Acción y efecto de validar.

***Validar*:** Dar fuerza o firmeza a algo, hacerlo válido.

---W---

Web Services: Es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

WebSphere Studio: Ofrece un entorno con soporte para casi todas las tecnologías, si bien está especialmente centrado en lo que parece ser una de las piedras filosofales del Web, los Applets de servidor o servlets.