Universidad de las Ciencias Informáticas

Facultad 3

Título: “Complemento de Configuración para la gestión de pruebas de carga y estrés a sistemas de gestión web con la utilización de la herramienta automatizada JMeter”.

Trabajo de Diploma para optar por el título de
Ingeniero en ciencias Informáticas

Autor:

Susell Fernández Pérez

Tutor:

Ing. Giselle Almeida González

Co-tutor:

Ing. Amírka Palacio Macías

La Habana, Junio 2012

Año del 54 aniversario de la Revolución

"Aunque haga muchos experimentos, mi hipótesis no queda confirmada, pero basta un solo experimento para confirmar mi error"

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser autor 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 29 días del mes de junio del año 2012

Firma del Autor	Firma del Tutor	Firma del Co-Tutor
Susell Fernández Pérez	Ing. Giselle Almeida González	Ing. Amírka Palacio Macías

DATOS DE CONTACTO

Autor: Susell Fernández Pérez

Universidad de las Ciencias Informáticas, C. Habana, Cuba
Correo electrónico: sfperez@estudiantes.uci.cu

Síntesis del tutor: Ing. Giselle Almeida González

Graduada de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. A partir de ese momento se incorpora como Analista en la línea de Costos y Procesos en el proyecto ERP Cuba. Actualmente se desempeña como Especialista de Calidad en la línea de Calidad en el mismo proyecto. En su trabajo de diploma identificó, modeló y realizó propuesta de mejoras de los procesos de negocio que se desarrollaban en el laboratorio de Calisoft. Ha sido tribunal de tesis y ha impartido clases de Contabilidad y Finanzas, Ingeniería de Software y actualmente imparte clases de Práctica Profesional 3.
Categoría docente: Instructor recién graduado.
Especialista de la Dirección de Calidad de Software.
Correo electrónico: galmeida@uci.cu

Síntesis del Co-tutor: Ing. Amílkar Gutiérrez Macías

Graduada de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2010. A partir de ese momento se incorpora como Especialista de Calidad en la línea de gestión de la Calidad en la UCID. Actualmente se desempeña en ese mismo rol en la línea de Calidad en el centro CEIGE. En su trabajo de diploma identificó, modeló y diseñó propuestas de mejoras de los modelos de organización de bases de casos en sistemas basados en conocimientos.
Especialista de la Dirección de Calidad de Software.
Correo electrónico: amacia@uci.cu

Agradecimientos

La realización del presente trabajo diploma implicó mucho esfuerzo y dedicación, no hubiera sido posible sin el apoyo un grupo de personas en especial a mi figurita ***Kiuver*** *y mi papá* ***Alien****.*

Agradecer a mi ***mamita*** *y mi* ***papito*** *por ser mis guías de toda la vida, por enseñarme que con trabajo y esfuerzo podemos lograr todas nuestras metas, por ser la razón que me impulsa a seguir adelante y que hoy quiera convertir todo su esfuerzo y sacrificio en este título de ingeniera.*

Agradecer a mis primas hermanitas ***Tita*** *y* ***Sol****, mi hermanito* ***Dito*** *del que siempre estoy muy orgullosa, mis mamis tías, mi padrasto* ***Cuza*** *por poder contar siempre con él y hacer feliz a mi mamita.*

A una gran historia gratos recuerdos que guardo de esta universidad…recuerdos que duran para toda la vida.

A mis tíos, tías y primos habaneros que me han querido y ayudado muchísimo.

A mis primeras amigas ***Ana****,* ***Daymra****,* ***Yisel****,* ***Lisandra****,* ***Anyelis****,* ***Katia*** *a las que conocí último pero bien especiales* ***Magui****, a todas las muchachitas del grupo.*

A todas estas buenas amistades que he encontrado en la universidad.

A mi tutora por estar en todo momento como profesional y amiga.

Agradecer a todos los que se preocuparon y los que atentos al desarrollo de la tesis aportaron su granito de arena, en fin gracias a todos por apoyarme a lograr mis sueños.

Dedicatoria

"La vida es comparable con un viaje en tren, comparación interesante porque nuestra vida está llena de embarques y desembarques, de accidentes, de sorpresas agradables, con subidas y bajadas tristes. Cuando nacemos, subimos y durante la travesía encontramos seres queridos, hermanos, amigos y amores. Muchos solo realizarán un corto paseo, otros estarán siempre a nuestro lado compartiendo alegrías, tristezas y recuerdos imborrables. Sentiré añoranzas, dejar a mis amigos será muy triste. Tengo la esperanza de que en algún momento nos encontremos en la estación principal, y tendré la emoción de verlos llegar con mucha más experiencia de la que tenían al iniciar el viaje."

Dedico mi tesis a las personas que más quiero en la vida, las que son mi fuente de inspiración y merecedoras de mi admiración y respeto.

A mi tío ***Manuel*** *que no se encuentra entre nosotros pero ejemplo de perseverancia.*

A lo más especial que me ha dado la vida…mis ***padres*** *por ser el centro y objetivo de mi existencia, por siempre confiar en mí.*

A mi hermanito querido ***Dito****, mi hermana* ***Tita*** *y a* ***Cuza*** *por ser cómplices de mis pocas travesuras y secretos. A toda mi gran* ***familia****…por llenarme de amor.*

A mis ***abuelitos*** *lindos por sus consejos y regaños.*

A mis buenos ***amigos*** *y amores que se convirtieron en mi familia estos 5 años llenándome de amor y felicidad.*

RESUMEN

Las pruebas de software son las encargadas de confirmar que la solución desarrollada cumple con las necesidades básicas o elementales, así como detectar faltas y fallos del mismo. Con la necesidad de lograr la excelencia se desarrollan diferentes tipos de pruebas de software, para las que se utilizan múltiples artefactos que permiten medir la calidad del producto.

La Universidad de las Ciencias Informáticas está estructurada por centros, entre éstos se encuentra el Centro de Informatización de la Gestión de Entidades donde se desarrollan diversas soluciones con el fin de gestionar y planificar los recursos en las entidades. Una de las desventajas que se presenta en el área y que influye directamente en la calidad del producto final es que no se tienen normados los tipos de pruebas que deben aplicarse para evaluar el requisito no funcional “rendimiento”. Existen múltiples herramientas automatizadas para el desarrollo de estas pruebas pero Apache JMeter es de las más completas para pruebas de carga y estrés.

En el presente trabajo diploma se propone una herramienta que permite aumentar la usabilidad del mismo para la realización de las pruebas de rendimiento de tipo carga y estrés. A partir de estas consideraciones se muestran los resultados, logrando validar los objetivos propuestos.

PALABRAS CLAVE: Calidad, Pruebas de carga y estrés, Complemento, JMeter, XML.

Índice de contenido

Índice de figuras

Índice de tablas

INTRODUCCIÓN

La industria de producción del software en el siglo XXI vive toda una revolución de cambios que han propiciado su constante evolución y mejores resultados en el desarrollo de productos. A pesar del auge alcanzado la mayoría de las soluciones propuestas no siempre llegan a satisfacer las necesidades del cliente en el plazo y el presupuesto pactados, pues enfocan su trabajo en reducir costos y tiempo descuidando la calidad del proceso de construcción y del producto [48].

La producción de aplicaciones informáticas es un negocio muy competitivo y Calidad del software es un concepto que se desarrolla junto a la industria, pues se debe ser capaz de superar las expectativas de cada cliente. Las pruebas de software son las encargadas de confirmar que la solución desarrollada cumple con las necesidades básicas o elementales, así como detectar faltas y fallos del mismo. Con la necesidad de lograr la excelencia se desarrollan diferentes tipos de pruebas de software, para las que se utilizan múltiples artefactos que permiten medir la calidad del producto.

La industria cubana está enmarcada en la necesidad de dominar e introducir en la práctica una profunda y eficiente gestión de la calidad. La Universidad de las Ciencias Informáticas (UCI) parte de dicha estrategia para apoyar el desarrollo del software y lograr insertarse en el mercado internacional con productos de una estructura óptima y eficiente. La UCI está estructurada por centros, entre estos se encuentra el Centro de Informatización de la Gestión de Entidades (CEIGE) donde se desarrollan diversas soluciones con el fin de gestionar y planificar los recursos en las entidades [9]. A estos sistemas solo se le realizan pruebas exploratorias, de sistema, regresión, liberación e internas; trayendo como consecuencia que en ocasiones las soluciones que se desarrollan en el centro no sean adaptables a los ambientes elegidos por el cliente. Estos resultados están debidos a que no se tiene normado los tipos de pruebas que deben aplicarse para evaluar el requisito no funcional rendimiento provocando desconocimiento al equipo de trabajo del tiempo máximo de respuesta permitido para una cantidad determinada de usuarios, la cantidad de usuarios que soporta conectados simultáneamente la aplicación y los tiempos de respuesta establecidos por cada petición que se haga al servidor.

Actualmente en el CEIGE las pruebas carga y estrés se realizan de manera manual, acción que requiere del empleo de gran cantidad de tiempo y recursos, por ello se ve afectada la calidad de los resultados de las mismas. Además no se ve afectada la precisión en cuanto a los resultados ya que el método manual puede traer consigo errores derivados del factor humano que no dispone de mucho tiempo para realizar este proceso.

Apache JMeter es una herramienta de pruebas, que permite medir el desempeño de aplicaciones Web, Base de datos y está desarrollada bajo los estándares de software libre. Estas pruebas realizan una correcta evaluación de la eficiencia del software, pero en el funcionamiento de la herramienta se han identificado algunas deficiencias:

- La grabación de los escenarios de prueba es trabajosa, debido a que la herramienta JMeter carga con todos los componentes del sistema en prueba incluyendo los elementos de respuesta, que para la ejecución de las pruebas de carga y estrés deben ser eliminados manualmente.
- Se hace difícil parametrizar valores para las aplicaciones debido a que no se especifica el tipo de datos que espera cada campo de los formularios que necesita la herramienta JMeter para ejecutar las pruebas de carga y estrés. Se suma a ello que la mayor fuerza de trabajo del Grupo de Aseguramiento de la Calidad de Software del CEIGE está compuesto por estudiantes poco capacitados para la gestión de pruebas de rendimiento y uso de esta herramienta.

Estas condiciones confirman que JMeter es poco práctica para realizar la mayoría de las actividades de configuración, grabado o de guardar la evidencia de las pruebas; haciéndose necesario para agilizar el trabajo y la calidad del proceso implementar un complemento de configuración al JMeter que permitirá realizar las pruebas en un ambiente más acogedor y fácil de entender.

Partiendo de lo mencionado anteriormente se plantea el siguiente **problema a resolver:** ¿Cómo perfeccionar las pruebas de carga y estrés para obtener una mayor usabilidad de la herramienta JMeter?

Siendo su **objeto de estudio:** Herramientas y procesos para pruebas de carga y estrés a sistemas de gestión web enmarcando su **campo de acción** en las pruebas de carga y estrés a sistemas de gestión web con la herramienta JMeter.

Para darle solución al problema planteado se define como **objetivo general:** Desarrollar un complemento de configuración para las pruebas de carga y estrés a sistemas de gestión web para obtener una mayor usabilidad de la herramienta JMeter.

Los **objetivos específicos:**

- Realizar el marco teórico- conceptual de la investigación.
- Implementar un complemento de configuración para las pruebas de carga y estrés a sistemas de gestión web que permita obtener una mayor usabilidad de la herramienta JMeter.

- Realizar pruebas con el objetivo de validar la solución propuesta.

Se tiene como **Idea a defender:** Si se desarrolla un complemento de configuración para la herramienta JMeter se mejorará la usabilidad de la misma para la realización de las pruebas de carga y estrés a sistemas de gestión web.

Para el correcto desarrollo del presente trabajo de diploma se emplearon métodos científicos de corte teórico y empírico tratando de mantener siempre un equilibrio entre lo cualitativo y lo cuantitativo.

Métodos Teóricos

Análisis – Síntesis: Para el estudio y análisis de la bibliografía, la formulación del problema y los objetivos, así como para la elaboración de las conclusiones y recomendaciones a fin de alcanzar los objetivos de la investigación.
Teórico Histórico – Lógico: Permitiendo hacer un estudio del comportamiento actual de la herramienta JMeter proyectando el análisis de la evolución histórica de los fenómenos, con la proyección lógica de su comportamiento futuro.

Métodos Empíricos

Entrevista: Se realizó una exhaustiva investigación donde se entrevistaron estudiantes y especialistas que aportaron importantes elementos de la investigación que permitieron analizar las deficiencias en el uso de la herramienta e identificar las necesidades existentes en la universidad para evaluar la eficiencia de los productos que se desarrollan.
Encuesta: A través de cuestionarios se identificaron requisitos funcionales y no funcionales, que permitieron elaborar una estructura a la solución acorde a las necesidades de los probadores. También realizar una evaluación de la preparación de los probadores en la gestión de pruebas de rendimiento y uso de la herramienta JMeter.

El presente trabajo está estructurado por 3 capítulos y anexos, en los cuales se dará respuesta a cada uno de los objetivos trazados en la investigación y se obtendrán resultados prácticos mediante la aplicación de pruebas de rendimiento que validarán los elementos propuestos.

Capítulo 1: Fundamentación teórica

Se realiza un estudio sobre las pruebas de software, su definición e importancia. Se investiga acerca del estado del arte del tema a desarrollar, su necesidad dentro de la herramienta JMeter, y por último un análisis de las tecnologías, y metodologías seleccionadas.

Capítulo 2: Características, análisis y diseño del sistema

Se presenta la propuesta de solución que se quiere implementar, se muestra el entorno donde se desarrolla el complemento mediante el modelo de dominio. Se elaboran los artefactos de esta fase: Diagrama de arquitectura, Diagrama de paquetes, Patrones de diseño, Diagrama de clases del diseño, Descripción de las clases, Historia de Usuarios (HU), Descripción de las HU y el Prototipo no funcional. Concluyendo el capítulo con el análisis de la arquitectura modelo n-capas y los diferentes patrones de diseño que se aplican en el desarrollo.

Capítulo 3: Implementación y pruebas

Se explica todo lo relacionado con la implementación, se elaboran los diagramas de despliegue y de componente y se muestran distintos elementos del desarrollo del complemento de configuración. Además se elaboran los casos de pruebas que se utilizan para realizar las pruebas de Unidad, Integración, Aceptación y el uso de listas de chequeo para evaluar la usabilidad del complemento.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se realiza un estudio de los conceptos de pruebas de software, el desarrollo y utilidad a nivel nacional e internacional. Seguidamente se desarrolla una comparación de las herramientas, lenguajes y metodologías que se utilizan dejando seleccionada la más óptima para darle solución a la problemática descrita.

1.2 Calidad del Software

Una empresa de software debe tener presente aspectos importantes de la calidad, todos ellos en función de las expectativas del cliente. La rapidez de la implementación del software y su eficacia son aspectos de gran relevancia para desarrollar productos informáticos.

La calidad de software como concepto puede verse desde diferentes puntos de vista; a continuación se aluden algunos de los definidos por diferentes autores:

CMMI1 plantea que la calidad es “la habilidad de un conjunto de características inherentes de un producto, componente de producto o proceso, para satisfacer los requerimientos del cliente”.

La norma internacional ISO2 9000:2000 que establece el vocabulario para los sistemas de gestión de la calidad define a esta como “grado en el que un conjunto de características inherentes cumple con los requisitos”.

La calidad del software es una compleja mezcla de factores que variarán a través de diferentes aplicaciones y según los clientes que las pidan [3].

[1] **CMMI:** (Capability Maturity Model Integration) Modelo para la mejora o evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. Desarrollado por el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon (SEI), y publicado en su primera versión en enero de 2002.

[2] **ISO:** (International Organization for Standardization) Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

Es el desarrollo de software basado en estándares con la funcionalidad y rendimiento total que satisfacen los requerimientos del cliente [17].

Después del análisis se concluye que la calidad no es un atributo que se le agrega al software, se desarrolla durante todo su ciclo de vida, y es quién define y evalúa el grado de cumplimiento de los requerimientos del sistema y por ende las exigencias del cliente.

1.3 Pruebas de software

La fase de pruebas es una de las más costosas del ciclo de vida software. En sentido estricto, deben realizarse pruebas de todos los artefactos generados durante la construcción de un producto, lo que incluye especificaciones de requisitos, casos de uso, diagramas de diversos tipos y, por supuesto, el código fuente y el resto de productos que forman parte de la aplicación [49].

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) define las pruebas de software como "[…] una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, donde se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente" [6].

Según Pressman el proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos [3].

Después de un estudio de diferentes bibliografías la autora del trabajo diploma ante los efectos de la investigación concluye que el proceso de pruebas permite evaluar, verificar e identificar la correspondencia con los estándares y procedimientos definidos por el centro que guie su desarrollo. El objetivo de estas se centra en identificar errores mediante el trabajo conjunto para evitar grandes pérdidas de recursos y tiempo.

1.3.1 Objetivos de las Pruebas al software

Los objetivos de las pruebas no sólo tienen que ver con corregir los errores, sino también con prevenirlos, influyendo y controlando el diseño y desarrollo del software. Las pruebas deben ser empleadas como modelos de los requerimientos de la aplicación que se ha de construir; por tanto, en las especificaciones de software deben incluirse especificaciones de pruebas.

Roger S. Pressman establece normas que pueden servir para los objetivos de las pruebas [3]:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- La prueba tiene éxito si se descubre un error no detectado hasta entonces.
- Encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles.
- Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.
- No deben ser redundantes.
- Una buena prueba no debe ser ni muy sencilla ni excesivamente compleja: es mejor realizar cada prueba de forma separada si se quiere probar diferentes casos.

Por lo tanto identificar cuáles son los objetivos ayuda a llevar un proceso bien organizado, identificar las prioridades de los clientes y realizar una evaluación certera.

1.3.2 Importancia de las Pruebas al software

Las pruebas al software constituyen un elemento de garantía a la calidad del producto y una revisión de los requisitos, el diseño y del código. Constituyen procesos que permite verificar y revelar la calidad de un producto software, utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador.

Verificación: El proceso de evaluación de un sistema o de uno de sus componentes para determinar si los productos de una fase dada satisfacen las condiciones impuestas al comienzo de dicha fase [22].

Validación: El proceso de evaluación de un sistema o de uno de sus componentes durante o al final del proceso de desarrollo para determinar si satisface los requisitos especificados [22].

El proceso de pruebas se debe hacer paralelo al proceso de construcción de construcción del software, permitiendo una constante comprobación del cumplimento de los requisitos planteados por el cliente. Con el uso de las diferentes pruebas se logra evitar impactos negativos como la pérdida de recursos y esfuerzo de trabajo, obteniendo experiencia y reconocimiento para el equipo de trabajo.

1.4 Métodos de prueba

Para la ejecución de las pruebas se deben llevar a cabo un conjunto de actividades con el objetivo de desarrollarlas en el menor tiempo posible, con la mayor calidad y de manera correcta para obtener resultados satisfactorios. Para lograr encontrar la mayor cantidad de errores y realizar una evaluación profunda de la solución se desarrollan 2 métodos de prueba:

Caja negra: pruebas funcionales sin acceso al código fuente de las soluciones, se trabaja con entradas y salidas. En la prueba de la caja negra, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta [56].

Caja blanca: pruebas con acceso al código fuente (datos y lógica). Se trabaja con entradas, salidas y conocimiento interno. En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y examinado el estado del programa en varios puntos [56].

1.4.1 Niveles de prueba

Las pruebas se desarrollan para evaluar para diferentes elementos del software, en disímiles escenarios o niveles de trabajo. Cada una de las pruebas se realiza en determinados momentos del ciclo de vida del software. Por lo explicado anteriormente se agrupan por niveles de acuerdo con las diferentes etapas del proceso de desarrollo:

Pruebas a nivel de Unidad

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Se encargan de un único caso a la vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto [20].

Pruebas a nivel de Aceptación

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento [21].

Pruebas a nivel de Integración

Estas pruebas son realizadas una vez que terminan las pruebas de unidad y preceden a la integración del sistema. Algunas veces llamadas integración y testeo, se basan principalmente en combinar y testear componentes individuales como un grupo, para corregir errores que pueden surgir a partir de la integración de esos componentes. No son verdaderamente pruebas de sistema, porque los componentes no están implementados en el ambiente operativo [18].

Prueba a nivel de Sistema

La prueba de sistema generalmente está constituida por una serie de pruebas diferentes cuyo propósito fundamental es ejercitar el sistema. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas [3].

Entre los tipos de pruebas más importantes en este nivel se encuentran:

- **Funcionales**: Las prueba funcionales están desarrolladas bajo la perspectiva del usuario, confirmando que el sistema hace lo que los usuarios esperan que haga, es decir, que se cumplan satisfactoriamente los requisitos funcionales.
- **De Usabilidad**: Se intenta determinar si el sistema será fácil de utilizar para sus usuarios.
- **De comunicación**: Se prueban las interfaces de comunicación entre diferentes componentes sistema.
- **De rendimiento**: Prueba el rendimiento del software en tiempo de ejecución. Determina si los tiempos de respuesta, tanto en condiciones normales como en condiciones especiales, se encuentran dentro de los límites predefinidos.

1.4.2 Tipos de pruebas de rendimiento

Se tiene como definición que las pruebas de rendimiento es el “grado en que un sistema o componente realiza sus funciones designadas dentro de las limitaciones dadas, tales como la velocidad, precisión o el uso de la memoria” [23].

Existen diferentes tipos de pruebas de rendimiento cuyo objetivo es observar la respuesta de la solución a la que se le está desarrollando las pruebas bajo diferentes escenarios, lo que proporciona los datos necesarios para saber si el sistema cumple con los requisitos de rendimiento o qué partes hacen que rinda de forma incorrecta. Cada una de estas pruebas proporciona resultados de acuerdo a objetivos específicos:

Pruebas de carga:

Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga [24].

Prueba de estrés:

Estas pruebas 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. Tienen como finalidad el determinar la robustez de una aplicación cuando la carga es extrema y ayuda a los administradores a determinar si la aplicación se comportará debidamente ante diferentes situaciones [24].

Prueba de estabilidad (soaktesting):

Esta prueba se realiza para determinar si la aplicación puede aguantar varias pruebas de carga seguidas o sea que se pruebe continuamente según el total de carga que soporte el sistema. Busca posibles deterioros o degradaciones en el rendimiento del sistema [25].

Pruebas de picos (spiketesting):

Esta prueba observa el comportamiento del sistema variando el número de usuarios concurrentes que se le introducen en forma de pico, o sea, se iría variando gradualmente la carga en determinados momentos de esta prueba [26].

Después del estudio de las diferentes bibliografías la autora de la investigación concluye que centrara los objetivos de la misma a las pruebas de carga y estrés dado que la necesidad urgente del centro conduce a lograr desarrollar en las aplicaciones una estructura que responda de manera eficaz en cualquier entorno de trabajo. En resumen estos 2 tipos de pruebas de rendimiento permiten realizar una evaluación en profundidad de la eficiencia del software en cuanto a carga de trabajo e identificar limitaciones, estos datos pueden ser utilizados durante el proceso de implementación para evitar grandes pérdidas de tiempo, presupuesto y prestigio para el centro.

1.5 Herramientas automatizadas para ejecutar pruebas de Rendimiento

Al existir diferentes tipos de pruebas de rendimiento, las cuales son muy útiles, siempre y cuando sean usadas a través de herramientas que permitan visualizar los detalles técnicos, gráficos altamente informativos, el grabado de los escenarios y resultados de las pruebas [51].

En la actualidad el uso de herramientas para la ejecución de las pruebas es necesario y de vital importancia para el flujo de pruebas en el desarrollo de software. Por lo que se han desarrollado diferentes soluciones con el fin de suplir las necesidades de las empresas.

El Laboratorio Industrial de Pruebas de Software (LIPS), perteneciente al Centro Nacional de Calidad de Software (CALISOFT) encargado de realizar pruebas de rendimiento a aplicaciones cliente/servidor. Para la realización de las mismas utilizan una herramienta muy eficiente JMeter.

Actualmente en el CEIGE las pruebas carga y estrés se realizan de manera manual y la acción de probar el software requiere del empleo de gran cantidad de tiempo y recursos, por ello se ve afectada la calidad de los resultados de las mismas. Al mismo tiempo de contar con problema relacionados con la precisión en cuanto a los resultados ya que el método manual puede traer consigo errores derivados del factor humano. A continuación un análisis de las herramientas automatizadas para la gestión de pruebas que permita identificar datos significativos para aumentar el nivel de usabilidad del JMeter durante la grabación y configuración de las pruebas a desarrollar con la herramienta.

1.5.1 LoadRunner

Es una herramienta para realizar pruebas rendimiento desarrollada originalmente por Mercury Interactive Corporation, que permite probar y analizar el comportamiento en condiciones normales, de estrés o de forma prolongada. Posee un sistema multiusuario y es una herramienta para pruebas de servidor.

Está estructurada por cuatro componentes el VUGEN, controlador, generador de carga y análisis, de manera general el VUGEN (Virtual UserGENerator) permite generar scripts que simularan un usuario real, el controlador permite agrupar y organizar los scripts. Como pieza principal el generador de carga recibe instrucciones del controlador y ejecuta scripts según la configuración realizada, y el análisis permite granular y comparar la información sobre los datos obtenidos durante la ejecución [37].

Crea archivos de registro mientras trabaja. En estos archivos de registro el usuario puede encontrar información sobre cualquier problema que se haya producido. Puede definir rastreos para:

- El motor
- El gestor de nivel medio
- La GUI de Web
- El módulo RIM (RDBMS Interface Module)

Los archivos de registro para grabación y reproducción sólo se crean en las máquinas donde se han iniciado recopilaciones de datos de la simulación de Grabación y reproducción. Los archivos de registro que se crean son tapm_playback_session.log y tapm_playback_driver.log. Estos pueden volverse a utilizar para monitorizar la aplicación una vez terminada su implantación.

1.5.2 QALoad

Herramienta de automatización de pruebas de carga para Web, Java, .NET, aplicaciones de planificación de recursos empresariales (ERP), aplicaciones de gestión de relaciones con los clientes y ambientes distribuidos (CRM). QALoad simula miles de usuarios desempeñando transacciones de negocio clave de la aplicación y permite validar que el sistema cumple aceptablemente con los niveles de servicio.

QALoad ofrece módulos configurables que facilitan el desarrollo de los archivos de prueba. Estos módulos, llamados EasyScripts, permiten a los ingenieros de pruebas grabar el tráfico que genera la aplicación y convertirlo en un archivo de prueba. Utiliza un lenguaje de scripting EasyScript para aumentar la productividad de los especialistas en pruebas de rendimiento.

Los archivos de prueba resultantes reflejan el tráfico real generado por la aplicación y miden el tiempo empleado para completar las transacciones para garantizar que el sistema que sometido a las pruebas alcanza las especificaciones para las que ha sido construido.

En el estudio de la herramienta se identificó que no cumple con las características necesarias para darle cumplimiento a los objetivos del presente trabajo. Se concluye que solo desarrolla pruebas de carga y los objetivos están centrados en desarrollar pruebas de carga y estrés.

1.5.3 Application Center Test

Centro de aplicaciones de prueba (ACT según sus siglas en inglés) es una herramienta de Microsoft incluida en Visual Studio .NET [44]:

- Application Center Test Enterprise Edition: esta versión está incluida en Application Center 2000 y se recomienda para pruebas avanzadas de esfuerzo de aplicaciones Web a gran escala.
- Application Center Test Developer Edition: esta versión está incluida en Visual Studio .NET Enterprise Developer y se recomienda para optimizar código durante el proceso de desarrollo. Puede utilizar la Developer Edition directamente dentro del Entorno de desarrollo integrado (IDE) de Visual Studio .NET, aunque con unas opciones limitadas de configuración de proyectos. Las opciones avanzadas de configuración de proyecto están disponibles cuando abre el proyecto dentro de un programa ACT independiente.

Diseñada especialmente para desarrollar pruebas de carga y estrés, permitiendo obtener toda la información necesaria para detectar problemas de rendimiento y escalabilidad en las aplicaciones. Además permite realizar pruebas funcionales gracias a las pruebas dinámicas, su funcionamiento reside en simular un gran número de usuarios abriendo múltiples conexiones al servidor y enviando peticiones HTTP [44].

Permite crear pruebas estáticas al importar archivos de registro de Internet Information Server (IIS) y soporta la creación de pruebas al registrar una sesión de explorador de Internet Explorer, archivos que pueden ser modificados utilizando diferentes comandos [44].

El objetivo principal de Application Center Test son las pruebas de nivel de carga de larga duración y carga alta, las pruebas dinámicas programables también pueden ser útiles en pruebas funcionales.

Ante los términos de la investigación presenta como desventaja que la herramienta se encuentra incluida en Visual Studio IDE que soporta varios de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET; y las soluciones web que se desarrollan en el CEIGE no son compatibles con el este. De igual manera Atentando negativamente que es un software privativo y en la UCI no existe licencia para su uso.

1.5.4 JMeter Apache 4.0

Herramienta Java dentro del proyecto de Jakarta, que permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web y bases de datos. JMeter se destaca por su versatilidad, estabilidad, por ser de uso gratuito, permite realizar pruebas web clásicas, incluye test de FTP, JDBC, JNDI, LDAP, SOAP/XML-RPC y Web Service (en Beta). También permite la ejecución de pruebas distribuidas entre distintos ordenadores para realizar pruebas de rendimiento [13].

El JMeter muestra los resultados de las pruebas en una amplia variedad de informes y gráficas. Además facilita una rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo [14].

Ventajas de la herramienta JMeter:

- De las herramientas gratis, es la más completa y útil para el tipo de pruebas en cuestión.
- Es una herramienta que sirve para realizar pruebas funcionales, pero también sirve para realizar pruebas de regresión en aplicaciones web, algo que, a veces es verdaderamente complicado, según la aplicación, pero que es casi imprescindible en el mantenimiento y evolución de las aplicaciones, si se quiere asegurar un nivel de capacidad adecuado en la entrega del producto.
- Tiene una estructura en árbol que le da potencia, permitiendo que sea la imaginación de quien la use la que ponga los límites a la hora de diseñar el plan de prueba. Y brinda mayor cantidad de variantes para recoger los resultados obtenidos que el resto de las herramientas gratis, lo que permiten hacer un análisis exhaustivo de las pruebas realizadas [13].

Trabaja bajo Unix (Solaris, Linux, etc.) y Windows (98, NT, 2000). (Barrios, 2007) [28].

En cuanto a la licencia para usar JMeter: Costo de adquisición: $0, curva de aprendizaje: Muy corta, Soporte: Manual completo, foros, y gran contenido en online [27].

La herramienta seleccionada para esta investigación es el JMeter, teniendo en cuanta que se destaca por su versatilidad y estabilidad dentro de las herramientas de libre distribución. Además facilita a una rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo. Presenta una estructura en árbol que le da potencia, permitiendo que sea la imaginación de quien la use la que ponga los límites a la hora de diseñar el plan de prueba, brindando mayor cantidad de variantes para recoger los resultados obtenidos, que el resto de las herramientas de libre distribución, lo que permite hacer un análisis exhaustivo de las pruebas realizadas.

La utilización del JMeter supone un 95 % de tiempo menos para la realización de estas pruebas. La mayor inversión de tiempo que se necesita para la realización de las pruebas es la fase de estudio de los casos de uso críticos en la aplicación Web y la elaboración del plan de pruebas en la herramienta. Permite almacenar los resultados de la prueba y se generar gráficos que representan los aspectos que se han probado. La posibilidad de contar con material para el estudio de la utilización de la herramienta, supone una garantía en el éxito de la ejecución de las pruebas y de la realización de un manual que permita a otros proyectos de gestión, identificar puntos de control para la elaboración de estas pruebas y la ejecución de las mismas además de la recopilación y análisis de los errores encontrados. Otro aspecto a favor de esta herramienta es su desarrollo en el marco del software libre, política que apoya la universidad y que por las características existe un por ciento superior de confianza en la utilización de la misma ya que se puede contar con todo el código que genera a la herramienta. Como desventaja de la herramienta se encuentra el alto consumo de rendimiento del CPU ya que es necesario para su mejor desempeño más de 512mb de RAM. Para una mejor utilización del rendimiento del CPU es necesario utilizar la herramienta Ant, libre y gratuita, que facilita la ejecución de estos test de manera útil pero la utilización de la misma supone restricciones en cuanto a la graficación de los resultados y acciones directas que se pueden realizar en la ejecución de estos test en la herramienta JMeter [51].

1.6 Metodologías de desarrollo de software

Metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información [7].

Existen gran cantidad de metodologías desarrolladas que se diferencian por su fortaleza o diferentes enfoques que provocan debilidades prácticas. Por lo tanto es de vital importancia en el desarrollo y ciclo de vida de los productos estructurar el trabajo según las metodologías obteniendo mayor organización y calidad

del proceso. Además de enfocar el trabajo a minimizar los riesgos y el ahorro de los recursos. En la investigación se realizó un estudio de las metodologías más fuertes y eficientes para resolver el problema planteado de acuerdo a las necesidades y características definidas para el desarrollo del complemento.

1.6.1 Rational Unified Process (RUP)

RUP es una metodología que proporciona una guía para dirigir las actividades de desarrollo unida al Lenguaje Unificado de Modelado (UML). En lo referente a sus características está que es dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

Está basada en 6 principios básicos:

- Adaptar el proceso a las necesidades del cliente.
- Equilibrar prioridades pues los requisitos de los diversos participantes pueden ser diferentes.
- Demostrar valor iterativamente, los proyectos se entregan de forma iterativa donde se analiza la estabilidad y calidad del producto, así como también los riesgos involucrados.
- Colaboración entre equipos.
- Elevar el nivel de abstracción determinado por el uso de conceptos reutilizables como patrón del software, lenguajes, frameworks, entre otros.
- Enfocarse en la calidad.

RUP es considerada la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

1.6.2 SXP

Metodología ágil conformada por sus referentes internacionales SCRUM y XP, diseñada pensando en proyectos de software libre. Brinda una estrategia tecnológica a partir de procedimientos ágiles, lo que permite el aumento de la creatividad y responsabilidad del personal implicado. Está conformada por SCRUM porque es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de XP el desarrollo que consiste en una programación rápida con el usuario final como parte del equipo.

Consta de 4 fases principales:

Planificación-Definición: establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.

Desarrollo: realiza la implementación del sistema hasta que esté listo para ser entregado; Entrega, puesta en marcha.
Entrega: puesta en marcha.

Mantenimiento: se realiza el soporte para el cliente.

De cada una de ellas se despliegan 7 flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implantación, prueba, entrega de la documentación, soporte e investigación, el cual se utiliza por el equipo de desarrollo cuando sea necesario, es decir, es un flujo que se puede mover y utilizarlo en cualquier parte del ciclo de vida del proyecto [29].

Se realizan actividades como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las HU, Diseño, Implementación, Pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, que permite mejorar el diseño cada vez que se añade una nueva funcionalidad [29].

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad [29].

Para el desarrollo SXP brinda una mayor cantidad de facilidades de organización y desempeño del trabajo. A diferencia de RUP que es más apropiada para proyectos grandes porque requiere de un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. SXP posee muchas semejanzas con las peculiaridades en el desarrollo del problema a resolver dado que es la metodología inteligente para equipos pequeños, con variabilidad en los requisitos, flexibilidad en el desarrollo y una entrega rápida de la solución.

1.7 Lenguajes de desarrollo y de modelado

1.7.1 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Existen muchas notaciones y métodos usados para el diseño orientado a objetos, ahora el personal sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

Los objetivos de UML son muchos pero se centra en expresar de forma gráfica el sistema que se desee implementar de una forma entendible y logra especificar cada una de sus características. A partir de los modelos especificados se logran construir los sistemas diseñados y el diseño se reutilizaría como parte de la documentación del producto.

Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML han propuesto su propia metodología de desarrollo, denominada el Proceso Unificado de Desarrollo [35].

Los aspectos que definen este Proceso Unificado son tres: es iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura [36]:

Dirigido por casos de uso: Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados.

Centrado en la arquitectura: En la arquitectura de la construcción, antes de construir un edificio éste se contempla desde varios puntos de vista: estructura, conducciones eléctricas y fontanería. Cada uno de estos aspectos está representado por un gráfico con su notación correspondiente. Siguiendo este ejemplo, el concepto de arquitectura de software incluye los aspectos estáticos y dinámicos más significativos del sistema.

Iterativo e incremental: Dividir un proyecto en varias fases y ciclos de vida en los que se realizan varios recorridos por todas las fases. Cada recorrido por las fases se denomina iteración en el proyecto en la que se realizan varios tipos de trabajo (denominados flujos). Además, cada iteración parte de la anterior incrementado o revisando la funcionalidad implementada. Se suele denominar proceso.

Para satisfacer los objetivos del problema planteado se define como lenguaje de modelado UML, centrándose las características relevantes anteriormente descritas. Permite desarrollar el trabajo fácil y

organizado, brindando muchísimas comodidades en el uso de la programación orientada a objetos y permite un fácil entendimiento entre el equipo de desarrollo y el cliente.

1.7.2 Lenguaje de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar, es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo [55].

Java

Lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. Se deriva en alto grado de C++, de tal forma que puede ser considerado como un C++ nuevo y modernizado [46].

Características del lenguaje:

- Es intrínsecamente orientado a objetos.
- Funciona perfectamente en red.
- Aprovecha características de la mayoría de los lenguajes modernos evitando sus inconvenientes. En particular los del C++.
- Tiene una gran funcionalidad gracias a sus librerías (clases).
- No tiene punteros manejables por el programador, aunque los maneja interna y transparentemente.
- El manejo de la memoria no es un problema, la gestiona el propio lenguaje y no el programador.
- Genera aplicaciones con pocos errores posibles.
- Incorpora Multi-Threading (para permitir la ejecución de tareas concurrentes dentro de un mismo programa).

JMeter es una herramienta desarrollada en Java por lo que se decide utilizar esta tecnología para evitar posibles errores o daños en la integración del desarrollo con la esta. Conocida tecnología multiplataforma por lo brinda la oportunidad de que el desarrollo pueda ser utilizado en Windows o Linux. Otro aspecto positivo en su uso es facilita el trabajo eliminando muchos de los errores comunes para los programadores, posee una arquitectura neutra, es portable y multitarea.

1.8 Herramientas de desarrollo

1.8.1 Herramienta CASE

Computer Aided Software Engineering (CASE) conocida como Ingeniería del software Asistida por computadoras, como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (Investigación, Análisis, Diseño, Implementación e Instalación) [30].

Visual Paradigm 6.4

Herramienta de software libre para el modelado visual UML que ayuda a la planificación, análisis y diseño, generación de código fuente y documentación de programas informáticos. Además permite representar todo tipo de diagramas en el ciclo de vida del desarrollo de software, también soporta estándares como Lenguaje de Modelado Unificado (UML), SysML, BPMN, XMI, entre otros [32].

Se caracteriza por:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia: gratuita y comercial.
- Soporta aplicaciones Web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversión (control de versiones).
- Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI.
- Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.

- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Generador de informes.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XML.

Para la modelación de cada uno de los procesos y artefactos que guiarán el trabajo del equipo de construcción del software se empleará Visual Paradigm para UML en su versión 6.4. Herramienta que utiliza UML como lenguaje de modelado, permite agilizar el trabajo ya que entre sus funcionalidades permite generar todos los diagramas de clases, código fuente desde diagramas y documentación en diferentes formatos; previendo que cuenta además con gran cantidad de documentación y demostraciones interactivas. En la UCI se cuenta con la licencia para su uso y tiene la característica de ser multiplataforma elemento de vital importancia debido a las políticas de migración definidas en la universidad.

1.9 Herramienta de desarrollo colaborativo

1.9.1 Control de versiones

Una versión, revisión o edición de un producto, es el estado en el que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas [53].

Subversion 1.6.12

Sistema de control de versiones fundado en 2000 por CollabNet para reemplazar el Sistema de Control de Versiones (CVS). Su fuerte radica en que los archivos versionados no tienen cada uno un número de revisión independiente y todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado. Es una herramienta de software libre bajo una licencia de tipo Apache/BSD y está preparado para funcionar en red.

Este sistema permite seguir la historia de los archivos y directorios a través de copias y renombrados, logra que las modificaciones a los archivos sean atómicas, creación de ramas y etiquetas, operación más eficiente y además logra enviar las diferencias en ambas direcciones. Se usa también integrado a Apache lo que permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.). Permite el soporte tanto de ficheros de texto como de binarios y puede ser servido mediante Apache, sobre WebDAV/DelatV permitiendo que clientes WebDAV utilicen Subversion de forma transparente [32].

Después del estudio de las diferentes bibliografías se seleccionó la herramienta Subversion de gran prestigio a escala internacional y eficiente para el versionado de productos. Permite tener un mejor control del código y con esto mayor aprovechamiento del el tiempo de desarrollo, debido a que disminuye los posibles retrasos causados por una mala administración de versiones y minimizamos los riesgos de pérdida de datos. Posibilita trabajar en distintas características o funciones simultáneamente, guardando los cambios en cada una de ellas y uniéndolos al desarrollo principal cuando estime conveniente la solución encontrada. Incluyendo que varias personas puedan modificar y administrar los mismos datos a pesar de encontrarse en lugares distantes una de las características de la herramienta que facilita el desarrollo en equipo. Otra de las excelencias que proporciona el que puedas volver a cualquier punto del desarrollo para ver qué aspecto tenía un determinado fichero de código, o volver a una versión anterior según la selección que se realice.

1.9.2 Entorno de desarrollo Integrado (IDE)

Un entorno de desarrollo integrado o IDE (en inglés: Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación que permite de forma cómoda y ágil editar, compilar, ejecutar y depurar programas [54].

NetBeans 6.9

Software de código abierto diseñado para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas y escrito en Java. Sun MicroSystems fundó este proyecto en junio 2000 y continúa siendo el patrocinador principal por el éxito que tiene entre los programadores [38].

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos por lo que pueden ser extendidas fácilmente por otros desarrolladores de software. Aunque está escrita en Java puede servir para cualquier otro lenguaje de

programación. Como existe un gran número de módulos para extender el NetBeans IDE, se puede decir que esta herramienta sea la única que se necesite a la hora de comenzar un proyecto [50].

Para el desarrollo se seleccionó al NetBeans en su versión 6.9 herramienta sumamente cómoda y de licencia gratis, usada para crear todo tipo de soluciones informáticas. Hace uso de la tecnología Java lenguaje definido para el desarrollo, dispone de soporte para creación de aplicaciones de escritorio con la herramienta de diseño Swing. Trae incluye soporte para la herramienta colaborativa Subversion de vital importancia para el control de cambios. En la selección se tiene en cuenta las políticas de migración del país y de la Universidad hacia el software libre.

1.10 Conclusiones

La investigación realizada profundiza en el concepto de calidad de software, los tipos de pruebas, las herramientas que se utilizan y el estado del arte permite llegar a la conclusión de que el grupo de Aseguramiento de la Calidad de centro CEIGE necesita realizar pruebas de rendimiento y para ello la elaboración de un complemento de configuración para agilizar el trabajo y además de lograr el desarrollo de las pruebas con eficiencia y profundidad. Además facilitará el trabajo con JMeter proporcionándole a los probadores una mayor práctica, experiencia y agilidad.

La metodología de desarrollo SXP se caracteriza por ser ágil, se centra en la interacción con el cliente, facilidades en la organización y desempeño del trabajo. Como herramienta para generar los diferentes artefactos el Visual Paradigm, herramienta multiplataforma que permite realizar todo tipo de diagramas y soporta el Lenguaje de Modelado Unificado (UML) elegido por ser potente y entendible que se utilizará para visualizar, especificar, construir y documentar el desarrollo del producto. Se seleccionó como IDE de desarrollo al NetBeans para que con todas estas especificaciones se diera respuesta a las necesidades planteadas por la problemática descrita. Para la selección de las herramientas y tecnologías se tuvo en cuenta las políticas de migración del país y de la Universidad hacia el software libre; por la flexibilidad, potencia, y múltiples posibilidades que este ofrece.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

El presente capítulo se enfocará en los detalles de la solución que se desea implementar para la herramienta JMeter, teniendo en cuenta las necesidades del centro CEIGE y del equipo de Aseguramiento de la Calidad. Incluye un análisis de las situación en el centro CEIGE y la universidad de manera general donde nace la necesidad de desarrollar pruebas de rendimiento. Se describirán todos los artefactos de esta fase: Diagrama de arquitectura, Diagrama de paquetes, Patrones de diseño, Diagrama de clases del diseño, Descripción de las clases, Historia de Usuarios (HU), Descripción de las HU y el Prototipo no funcional.

2.2 Descripción de la situación actual

En el centro CEIGE solo se desarrollan pruebas exploratorias, de sistema, regresión, liberación e internas lo que no garantiza que las soluciones cumplan con todos los elementos funcionales y no funcionales. Existen inconformidades por parte del cliente después de la entrega de los productos pues la eficiencia no fue evaluada por el equipo de desarrollo y el de Aseguramiento de la Calidad. Como una necesidad real de vital importancia es desarrollar pruebas de rendimiento, los beneficios de la interpretación de los resultados serán:

- Errores de rendimiento antes del despliegue de un producto.
- Respuesta del sistema en determinados ambientes de trabajo elegidos por el cliente.
- Tiempo máximo de respuesta es aceptado para una cantidad determinada de usuarios.
- Cuántos usuarios conectados simultáneamente soporta la aplicación.
- Los tiempos de respuesta por cada petición que se haga al servidor.

El desarrollo del complemento de configuración estará solucionando varios de los problemas prácticos de la herramienta JMeter para el desarrollo de las pruebas agilizando el proceso y apoyando la calidad de cada producto permitiendo la aplicación de estas pruebas.

2.3 Propuesta de sistema a desarrollar

Ante la necesidad del CEIGE se propone como solución una herramienta que apoye y facilite el trabajo con la herramienta JMeter para el desarrollo de pruebas de carga y estrés.

Específicamente se desarrollará un complemento de configuración que es descrito como una aplicación de escritorio que no se integrará al JMeter para generar un fichero de configuración con todos los formularios y muestreadores necesarios para el desarrollo de las pruebas, pero este hará uso del fichero para desarrollar las pruebas. El mismo facilitaría el trabajo del equipo de calidad y solucionaría algunos elementos que hacen poco práctica la herramienta JMeter como el grabado de las pruebas, la selección, parametrización y configuración de los formularios y muestreadores.

2.4 Modelo de dominio

Después de la investigación realizada para dar solución a la propuesta desarrollo de un complemento para la herramienta JMeter se propone el modelo de dominio, permite representar una secuencia lógica de conceptos lo más reales a las características físicas que posee el mismo, se utiliza para recoger y expresar el entendimiento obtenido del análisis paso que antecede al diseño de los sistemas.

La selección está enfocada al modelo de dominio debido a la poca estructuración de los procesos de negocio que provee el desarrollo del complemento para la herramienta JMeter. Sus objetivos están dados en lograr un entendimiento del contexto en que se sitúa el sistema, describir el funcionamiento de la herramienta mediante conceptos y relaciones

A continuación se presenta dicho modelo, muestra de todos los elementos o conceptos que representan y simbolizan objetos lo más semejante al mundo real involucrados con el negocio:

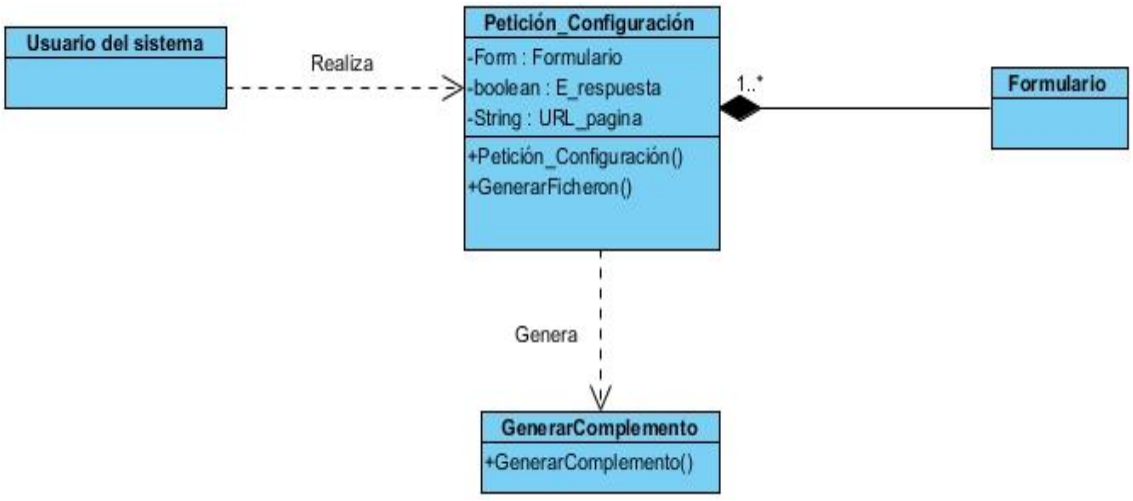

Figura 1.Modelo del dominio.

Usuario del sistema: el usuario del sistema será el probador, quien interactúa e introduce los datos de la aplicación a la que se le realizará las pruebas.
Formulario: representa cada uno de los formularios que son necesarios para la configuración de la pruebas.
E_respuesta: representa una opción para facilitar el grabado de la pruebas, permite decidir al usuario si desea cargar los elementos de respuesta de la aplicación.
URL_pagina: es para introducir la dirección URL de la web a la que se le realizarán las pruebas.

2.5 Requisitos del software

La ingeniería de requisitos es la encargada de recoger todos los detalles funcionales y no funcionales obtenidos de un extenso proceso de entendimiento con el cliente. La información recopilada debe traducirse de manera sencilla y bien entendible para que el desarrollo del producto sea parejo a las necesidades del cliente. Los requisitos funcionales definen el comportamiento interno de la aplicación, describen en detalle cada funcionalidad y los no funcionales complementan los funcionales detallando el ambiente a desarrollar o detalles técnicos.

La captura de requisitos es la actividad en la que el equipo de desarrollo de un sistema de software identifica las necesidades que debe cubrir el sistema.

2.5.1 Requisitos funcionales

RF 1 Gestionar URL de la aplicación.

RF 1.1: Añadir dirección de la aplicación. El sistema debe ser capaz de añadir tantas direcciones URL le indique el usuario.
RF 1.2: Eliminar dirección URL. El sistema debe permitir eliminar la dirección URL seleccionada por el usuario.
RF 1.3: Modificar dirección URL. El sistema debe permitir modificar la dirección URL seleccionada por el usuario.
RF 1.4: Listar dirección URL. El sistema debe listar la(s) dirección(s) URL añadida(s) por el usuario.
RF 2: Listar componentes de la aplicación. El sistema debe cargar todos los componentes de la aplicación y mostrarlos.

RF 3: Realizar configuración http de la pruebas. El sistema debe permitir al usuario definir los parámetros http con que desee desarrollar las pruebas.

RF 4: Gestionar formularios para el desarrollo de las pruebas. El sistema debe permitir seleccionar al usuario los formularios que desee para el desarrollo de la prueba.

RF 4.1: Añadir configuración de los formularios para las pruebas. El sistema debe permitir al usuario parametrizar los valores de cada uno de los formularios seleccionados para el desarrollo de la prueba.

RF 4.2: Modificar configuración de los formularios para las pruebas. El sistema debe permitir al usuario modificar la parametrización de cada uno de los formularios seleccionados para el desarrollo de la prueba.

RF 4.3: Eliminar formularios de pruebas. El sistema debe permitir al usuario desmarcar los formularios que no desee incluir en el desarrollo de la prueba.

RF 5: Generar fichero XML. El sistema debe generar un fichero XML para la configuración de la herramienta JMeter.

2.5.2 Requisitos no funcionales

Usabilidad:

Facilidad de uso por parte de los usuarios o probadores**:** el sistema debe presentar una interfaz amigable que permita una fácil interacción con él y llegar de manera rápida y efectiva a configurar y generar el archivo XML Además la interfaz debe permitir un manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación al mismo.

Facilidad de uso por parte de los usuarios o probadores**:** el sistema debe presentar una interfaz amigable que permita una fácil interacción con él y llegar de manera rápida y efectiva a configurar y generar el archivo

Especificación de la terminología utilizada: el sistema debe adaptarse al lenguaje y términos utilizados por los probadores con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.

Estándares de usabilidad:

- Hacer uso de gráficos e imágenes claras y nítidas.
- Respetar el espacio entre los enlaces y los botones.
- Evitar el uso de los menús en cascada.
- Proporcionar a los formularios un título que muestre claramente su función.
- Utilizar con los textos una nomenclatura clara y familiar.

- Hacer corresponder el tamaño visible de los campos de texto con la longitud del contenido que ha de introducir el usuario.
- Establecer en los campos, siempre que sea posible, una opción por defecto.
- Identificar claramente los campos obligatorios y opcionales.
- Informar al usuario el resultado de una acción cuando esta se ejecute.

Escalabilidad:

El sistema debe ser capaz de permitir la incorporación de nuevas funcionalidades adaptándose de manera natural a los procesos de gestión que se requieran.

Software:

Para ejecutar y correr el programa solo se necesita contar con una PC con la Máquina Virtual de Java instalada.

Requerimientos del diseño e implementación:

Como característica específica se utilizarán estas herramientas y metodologías en las fases de construcción del software.

- Se debe implementar en el lenguaje Java.
- El archivo que genera solo es utilizable para la herramienta JMeter.
- Para el análisis y el diseño se utilizará la metodología SXP, usando el lenguaje de modelación UML y como herramienta para llevar a cabo el modelado Visual Paradigm.
- Para la implementación del complemento se utilizará el IDE NetBeans, específicamente Swing una herramienta para el desarrollo de aplicaciones de escritorio.

2.6. Descripción de los actores del sistema a automatizar

Tabla 1.Actores del sistema.

Nombre del Actor	Descripción
Probador	Estos usuarios pueden configurar todos los datos necesarios para la gestión exitosa de las pruebas.

2.7 Descripción de las historias de usuarios

Se realiza una descripción de las Historias de Usuarios (HU) enlazadas a sus tareas ingenieriles y al responsable asignado para el cumplimiento de la misma. Ver anexo 1.

A continuación se muestran las HU más importantes del proceso:

Tabla 2.Historia del usuario 6. Realizar configuración http de la pruebas.

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Realizar configuración http de la pruebas.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Susell Fernández Pérez	**Iteración Asignada:** 1ra Iteración
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo:	**Puntos Reales:** 2
Descripción. El sistema debe permitir al usuario definir los parámetros http con que desee desarrollar las pruebas. El usuario debe llenar todos los campos del formulario, de forma obligatoria solo la cantidad de hilos a simular.	
Observaciones	
Prototipo de Interface:	

Tabla 3.Tarea de Ingeniería 1.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 6
Nombre Tarea: Almacenar en el fichero XML los datos entrados por el usuario correspondientes a la configuración http.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 16/04/2012	**Fecha Fin:** 29/04/2012
Programador Responsable: Susell Fernández Pérez	
Descripción: Se almacenan los datos con la estructura correspondiente a un árbol general dentro del templete base.	

Tabla 4.Historia del usuario 10. Generar fichero XML.

Historia de Usuario	
Número: 10	**Nombre Historia de Usuario:** Generar fichero XML.
Modificación de Historia de Usuario Número: Ninguno	
Usuario: Susell Fernández Pérez	**Iteración Asignada:** 2da Iteración
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semanas
Riesgo en Desarrollo: Alta	**Puntos Reales:** 1
Descripción: El sistema debe generar un fichero XML para la configuración de la herramienta JMeter. El sistema debe mostrar un mensaje confirmando que entradas fueron hechas satisfactoriamente, el usuario presiona el botón buscar para indicar la dirección donde desea guardar el archivo XML que se generará y luego se presiona el botón Generar para realizar la acción. Luego muestra un mensaje confirmando que realizo satisfactoriamente la acción y si	

desea abrir la carpeta contenedora.
Observaciones:
Prototipo de Interface: 

Tabla 5.Tarea de Ingeniería 2.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 10
Nombre Tarea: Investigar cómo se genera un archivo de extensión XML.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 14/05/2012	**Fecha Fin:** 20/05/2012
Programador Responsable: Susell Fernández Pérez	
Descripción: Se realizará un estudio de las diferentes vías para generar un archivo de extensión XML.	

Tabla 6.Tarea de Ingeniería 3.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 10
Nombre Tarea: Investigar cómo se genera un archivo de extensión XML.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 1

Fecha Inicio: 14/05/2012	**Fecha Fin:** 20/05/2012
Programador Responsable: Susell Fernández Pérez	
Descripción: Se realizará un estudio de las diferentes vías para generar un archivo de extensión XML.	

Tabla 7.Tarea de Ingeniería 4.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 10
Nombre Tarea: Estudiar funcionamiento y características de la herramienta JMeter para simular su funcionamiento al generar el archivo XML e interpretación de los datos.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 1
Fecha Inicio: 14/05/2012	**Fecha Fin:** 20/05/2012
Programador Responsable: Susell Fernández Pérez	
Descripción: Se realizará un estudio de la herramienta JMeter para simular su funcionamiento al generar el archivo e interpretar los datos.	

Tabla 8.Tarea de Ingeniería 5.

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 10
Nombre Tarea: Implementación de la funcionalidad generar archivo XML.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 14/05/2012	**Fecha Fin:** 20/05/2012
Programador Responsable: Susell Fernández Pérez	

Descripción: Se implementaron las funcionalidades necesarias para la creación del documento XML con la estructura adecuada para la integración con el JMeter.

2.8 Diseño del sistema

El diseño del sistema define todo el proceso que permite representar todos los aspectos a construir en el sistema, siendo la forma de materializar en detalle los requerimientos o necesidades del cliente.

2.8.1 Arquitectura del sistema

La IEEE define que “La arquitectura del software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el entorno, y los principios que dirigen su diseño y evolución” [39].

La arquitectura organiza todo el proyecto de trabajo a desarrollar, logrando una visión general y detallada de las características y relaciones de cada uno de los elementos que componen el software.

Para el desarrollo se ha decidido utilizar como estilo arquitectónico Modelo n-tier (n-capas) de informática distribuida, muy utilizado para la construcción de aplicaciones multiplataforma. Este modelo permite desarrollar de manera paralela en cada capa. De su uso se obtienen aplicaciones más sólidas debido al encapsulamiento. El mantenimiento y soporte se hace más sencillo, brinda mayor flexibilidad y principalmente una alta escalabilidad, proporcionando a la aplicación manejar gran cantidad de peticiones con el mismo rendimiento solo añadiendo más hardware. En la siguiente figura se muestra un diagrama con la relación existente entre las 2 capas que componen la arquitectura de la solución:

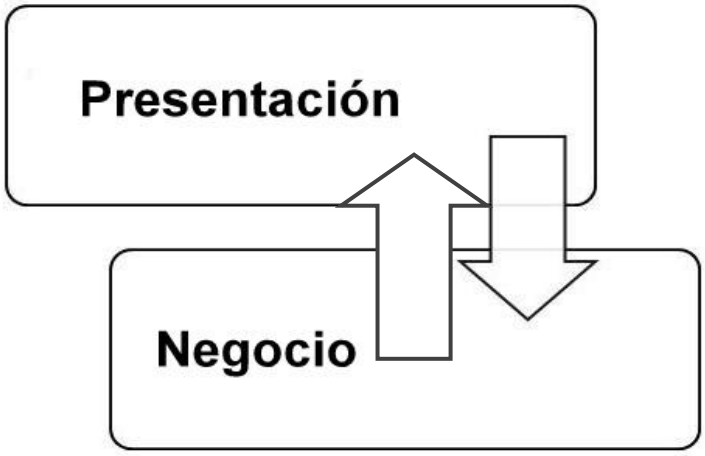

Figura 2.Arquitectura 2 capas.

El tipo de arquitectura de un sistemas de software está definido como su división en capas horizontales y particiones verticales, puede estar modelados e implementados en mas de 2 capas y no necesariamente los sistemas tienen estrictamente que contener un número especifico de capas. Debido a la estructuracion de los procesos y requisitos funcionales la arquitectura del complemento contiene la capa de presentación refiriendose a la interfaz o parte fisica del sistema y el negocio que contiene la lógica o implementación.

2.8.2 Diagrama de paquetes.

El diagrama de paquetes representa la estructura lógica de un sistema así como las dependencias o relaciones entre estas. En la figura 4 se muestra el diagrama de paquetes, cuyo contenido refleja la estructura y organización que tiene definido el complemento y el que está compuesto por los siguientes paquetes:

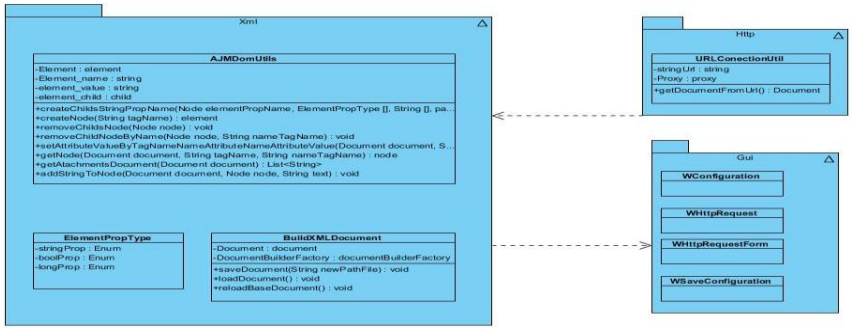

Figura 3.Diagrama de paquetes.

Gui: Contiene las clases visuales.

Xml: Constituye el paquete principal del complemento, contiene las clases principales AJMDomUtils encargada de proveer apoyo a la demás implementando funcionalidades auxiliares y BuildXMLDocument responsable de la organización de todos los datos para dar estructura y generar el archivo XML de manera que fuera integrable al JMeter.

Http: En su contenido tiene la clase URLConectionUtil que provee una de las funcionalidades del complemento, encargada de obtener la estructura o composición de la página a la que apunta determinado URL.

2.8.3 Patrones de diseño

Los Patrones GRASP describen cada uno de los principios principales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Experto: La asignación de una responsabilidad a la clase que cuenta con la información necesaria para cumplir una funcionalidad en específico nos permite tener un código correcto y funcional a la hora de utilizarlo. Este patrón se evidencia en la clase BuildXMLDocument pues tienen la responsabilidad de cargar y guardar el XML.

Creador: En el desarrollo para la proceso de creación u obtención del objeto Document se utilizó dicho patrón para las clases que tienen como responsabilidad crear instancias de otras para poder realizar completamente el funcionamiento del proceso. Este patrón se evidencia en algunas clases como Document, Element y BuildXMLDocument.

Bajo Acoplamiento: Se utilizó este patrón con la inyección de dependencias en las clases, las cuales son interfaces que poseen un comportamiento específico, el cual obtienen de las clases BuildXMLDocument y AJMDomUtils, posibilitando así asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases posible.

Alta cohesión: La relación de las clases como AJMDomUtils, BuildXMLDocument definen que generalmente una clase que este altamente cohesionada tiene bajo acoplamiento desde el punto de vista de clase ya que posee la menor cantidad de dependencias posibles con otras clases y a su vez posee responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.

Singleton: Define una instancia única para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Garantiza que una clase sólo tenga una instancia y proporcionar un punto

de acceso global a ella. Muestra que se evidencia en todo el desarrollo de la solución pues se trabaja con referencias de un mismo objeto Document en todas las clases.

2.8.4 Diagrama de clases

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas.

En la siguiente representación la organización de las clases está dada por paquetes definidos como se muestra en la figura 5:

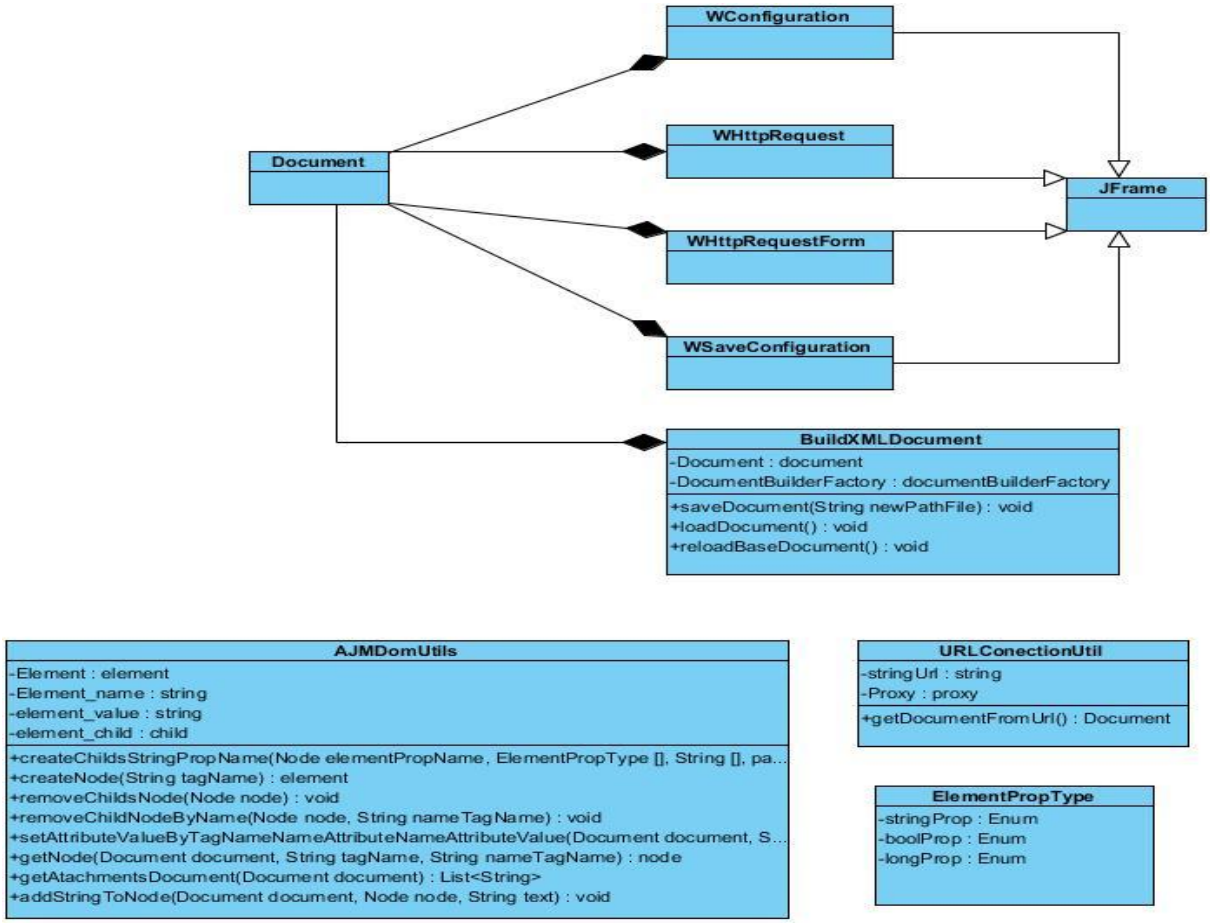

Figura 4.Diagrama de clases.

2.9 Estructura del script XML para pruebas de carga y estrés

Para el desarrollo de pruebas de carga y estrés la herramienta JMeter al poseer una estructura en árbol permite que cualquier parte o rama del árbol pueda ser guardada de forma independiente, para ser reutilizada en otras pruebas. Esta característica señala que el usuario del sistema puede incluir en su archivo de configuración los elementos que el estime conveniente.

Los elementos jerárquicos a configurar y definir para el plan de pruebas son los siguientes:

- Grupo de hilos
- Aserción de respuesta (afirmaciones), para comprobar la respuesta. Puede comprobarse el texto, o la URL, o el código de respuesta, o el mensaje de respuesta, e indicar si coincide con una serie de patrones, o no.
- Informe Agregado para ver los resultados de las pruebas.
- Respuesta Http, genera un caso de prueba es a través de una navegación de usuario. Para ello, se indica que JMeter actué como proxy, para capturar la navegación del caso de prueba.
- Ver árbol de resultados, muestra un árbol con todas las respuestas y sus tiempos.

La estructura general del archivo XML es un árbol lo que beneficia al probador a definir un plan de pruebas de más calidad apoyándose en su ingenio. La base está formada por un Banco de trabajo, un Plan de pruebas y demás elementos que se definan como se expuso anteriormente. A continuación se muestra una figura con la estructura base del archivo XML que genera el JMeter:

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="1.8">
    <hashTree>
        <WorkBench guiclass="WorkBenchGui" testclass="WorkBench" testname="Banco" enabled="true"/>
          <hashTree>
            <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Plan " enabled="true">
                <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Plan " enabled="true">
                    <stringProp name="TestPlan.comments"></stringProp>
                    <boolProp name="TestPlan.functional_mode">true</boolProp>
                    <boolProp name="TestPlan.serialize_threadgroups">true</boolProp>
                    <elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass
                        <collectionProp name="Arguments.arguments">
                        </collectionProp>
                    </elementProp>
                    <stringProp name="TestPlan.user_define_classpath">"Directorio"</stringProp>
                </TestPlan>
         <hashTree/>
     </hashTree>
 </jmeterTestPlan>
```

Figura 5.Estructura base del archivo XML generado por el JMeter.

2.10 Conclusiones

En este capítulo se realizó una descripción las características del complemento de configuración mediante los artefactos del análisis y diseño descritos anteriormente. En un análisis de la situación problémica se identificaron los elementos necesarios para desarrollar el proceso de construcción y una descripción del flujo del proceso de gestión de pruebas utilizando la herramienta JMeter. En el proceso se obtuvieron cuáles eran los procesos que se debían mejorar para lograr una mayor usabilidad de la herramienta, los actores, requerimientos funcionales y no funcionales descritos en Historias de usuario. A partir de todos los artefactos generados se puede dar paso al flujo implementación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

Como parte del desarrollo del complemento es de vital importancia y necesario para solidificar los objetivos propuestos la implementación y validación del sistema. Para alcanzar dicho propósito se modelará el diagrama de componentes y el prototipo funcional del complemento. Además, son realizadas las plantillas de las Pruebas de Aceptación para las Historias de Usuario implementadas.

3.2 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas, pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, etc. [41].
Permiten una sencilla representación de la organización y dependencia de los componentes, además que pueden ser usados para modelar y documentar la arquitectura de cualquier sistema. En la figura 6 se muestran los componentes de la solución:

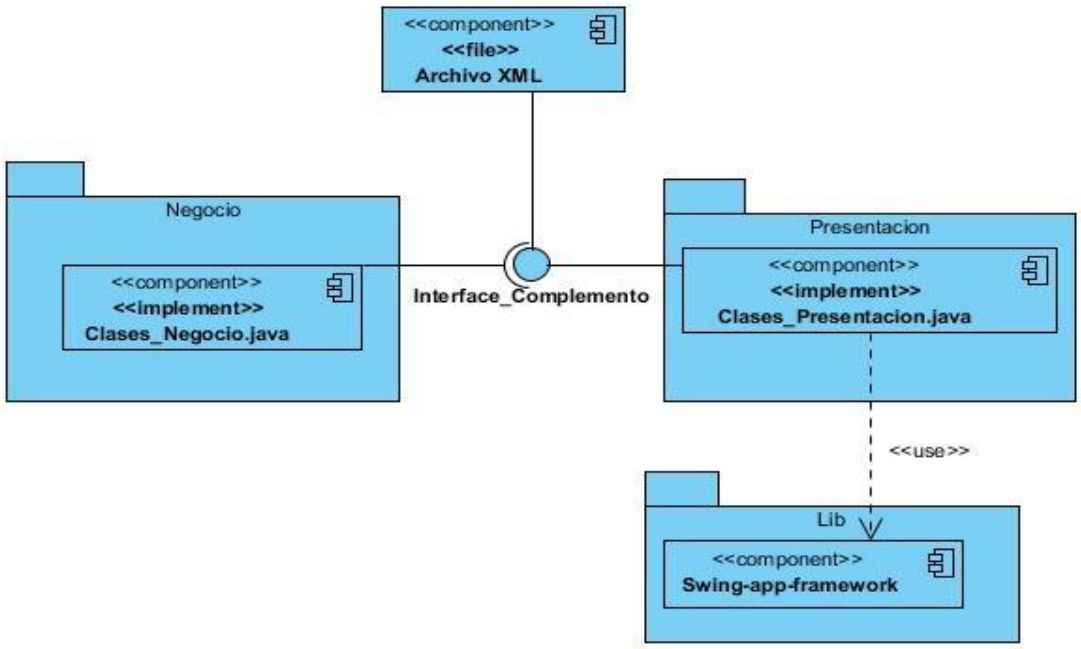

Figura 6.Diagrama de componentes.

A continuación se hace una breve descripción de los componentes del diagrama:

Archivo XML: archivo que genera el complemento de configuración que guarda los datos que fueron configurados durante el proceso ejecución del sistema.
Clases_Negocio: representación de las clases que se encuentran en la capa de negocio AJMDomUtils, ElementPropType, BuildXMLDocument y URLConectionUtil.
Clases_Presentacion: representación de las clases visuales perteneciente a la capa de presentación WConfiguration, WHttpRequest, WHttpRequestForm y WSaveConfiguration.
Swing-app-framework: librería para el diseño de la aplicación de escritorio que se fusiona con todas las clases visuales.

3.3 Descripción de los métodos y clases implementadas durante el desarrollo del complemento de configuración

Clase: BuildXMLDocument

En esta clase posee como atributo un Document que es utilizado por referencia en las formas visuales, las que lo modifican haciendo uso de la clase AJMDomUtils. Su objetivo se centra en cargar y guardar el documento XML que se desea generar; consta de 2 algoritmos principales:

```
public void saveDocument(String newPathFile) {
    try {
        FileOutputStream f = new FileOutputStream(newPathFile);
        TransformerFactory factory = TransformerFactory.newInstance();
        Transformer transformer = factory.newTransformer();
        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
        transformer.transform(new DOMSource(this.document), new StreamResult(f));
        f.close();
    } catch (IOException ex) {
        Logger.getLogger(BuildXMLDocument.class.getName()).log(Level.SEVERE, null, ex);
    } catch (TransformerException ex) {
        Logger.getLogger(BuildXMLDocument.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figura 7.Algoritmo implementado durante el desarrollo del complemento para guardar fichero XML.

```
public void loadDocument(String pathFile)
{
    try {
        try {
            this.document = documentBuilderFactory.newDocumentBuilder().parse(new FileInputStream(pathFile));
        } catch (ParserConfigurationException ex) {
            Logger.getLogger(BuildXMLDocument.class.getName()).log(Level.SEVERE, null, ex);
        }
    } catch (SAXException ex) {
        Logger.getLogger(BuildXMLDocument.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(BuildXMLDocument.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figura 8.Algoritmo implementado durante el desarrollo del complemento para cargar fichero XML.

Clase: AJMDomUtils.

Esta clase es la base de la estructura de XML y otras funcionalidades implementadas como auxiliar para otras clases. Define cómo se declara cada una de las etiquetas y variables definidas por la herramienta e identificada en un análisis previo.

```
public void createChildStringPropName(Node elementPropName, String name, String value, String metadata){
    Element elementName = this.createNode("stringProp");
    elementName.setAttribute("name", "Argument.name");
    elementName.setTextContent(name);

    Element elementValue = this.createNode("stringProp");
    elementValue.setAttribute("name", "Argument.value");
    elementValue.setTextContent(value);

    Element elementMetadata = this.createNode("stringProp");
    elementMetadata.setAttribute("name", "Argument.metadata");
    elementMetadata.setTextContent(value);

    elementPropName.appendChild(elementName);
    elementPropName.appendChild(elementValue);
    elementPropName.appendChild(elementMetadata);
}
```

Figura 9.Algoritmo implementado durante el desarrollo del complemento para definir etiquetas y variables acordes a las generadas por la herramienta JMeter.

3.4 Tipo de pruebas a desarrollar

La metodología seleccionada SXP utiliza las buenas prácticas de XP un desarrollo guiado por pruebas, posibilitando darle al cliente una idea de las verdaderas funcionalidades que tiene el producto. Mediante esta filosofía se reduce el número de errores no detectados, así como el tiempo entre la introducción de este en el sistema y su detección [29].Las pruebas se desarrollan durante toda la fase de construcción al final de cada iteración, las mismas están divididas en dos grupos unitarias y de aceptación. Dadas las características del software implementado se hace necesario realizar pruebas de integración pues la herramienta es la encargada de generar el fichero de configuración para que posteriormente sea utilizado por el JMeter.

Evaluación de la integración del complemento de configuración con el JMeter: se basan en unir y probar los componentes individuales como un grupo, para corregir errores que pueden surgir a partir de la integración de esos componentes. En el laboratorio de pruebas del centro se escogió una muestra de 3 probadores para generar el archivo con la herramienta y cargarlo al JMeter lo que resulta una evaluación de la integración. Las pruebas fueron realizadas con resultados satisfactorios como se muestra en la siguiente figura 10 JMeter cargó el archivo generado por el complemento y se utilizó para el desarrollo de pruebas en el laboratorio de pruebas del centro:

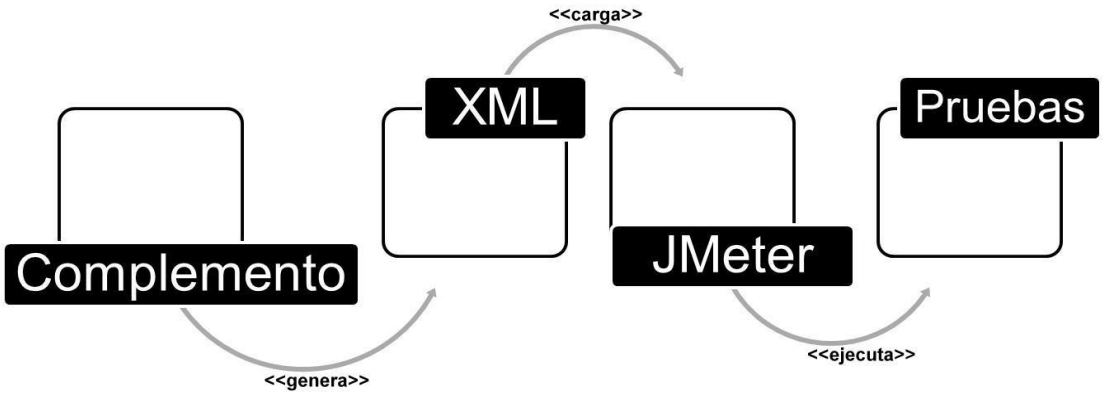

Figura 10.Gráfico resultado de la integración del complemento con la herramienta JMeter.

Pruebas de aceptación: realizan una evaluación al finalizar cada iteración comprobando si se cumplió con los requerimientos del cliente. En cada una de las iteraciones se traducen las Historias de usuarios a Pruebas de aceptación, las que permiten realizar una evaluación de todas las funcionalidades contra la descripción de cada historia. Ver anexo 2.

A continuación se revelan los diseños de caso de prueba de aceptación de algunos escenarios:

Tabla 9.Caso de Prueba de Aceptación HU6P6: Realizar configuración http de las pruebas.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU6P6	**Nombre Historia de Usuario:** Realizar configuración http de la pruebas.
Nombre de la persona que realiza la prueba: Ana Margarita Martínez Sarrión	
Descripción de la Prueba: Después de la configuración de cada uno de los formularios que el usuario desea incluir en sus pruebas realizar la configuración http es de obligatoriedad.	
Condiciones de Ejecución: Se ejecuta la aplicación satisfactoriamente.	
Entrada / Pasos de ejecución: El usuario debe llenar cada uno de los campos de manera obligatoria: - Nombre de la petición http. - Comentario de la petición http, que constituye una breve descripción de los objetivos de los datos que va a introducir. - Cantidad de hilos a simular, se traduce a la cantidad de usuarios que el sistema va a simular.	
Resultado Esperado: Los datos son guardados satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 10.Caso de Prueba de Aceptación HU7P7: Añadir configuración de los formularios para las pruebas.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU7P7	**Nombre Historia de Usuario:** Añadir configuración de los formularios para las pruebas.
Nombre de la persona que realiza la prueba: Ana Margarita Martínez Sarrión	
Descripción de la Prueba: El sistema debe permitir al usuario parametrizar los	

valores de cada uno de los formularios seleccionados para el desarrollo de la prueba.
Condiciones de Ejecución: El usuario debe habilitar al menos un formulario para continuar la ejecución de los pasos siguientes y debe llenar todos los campos con obligatoriedad.
Entrada / Pasos de ejecución: El usuario debe seleccionar la opción habilitar para que los datos del formularios sean almacenados. - Nombre del formulario. - Comentario del formulario, que constituye una breve descripción del objetivo con que se incluye para la ejecución de esta prueba. - Los campos de selección tienen valores por defecto, pero permiten selección de otros valores. - Los formularios que son muesteadores o tablas incluyen la opción de escoger una ruta para almacenar estos datos de manera independiente. Este campo debe llenarse de manera obligatoria.
Resultado Esperado: Habilitar los formularios seleccionados por el usuario.
Evaluación de la Prueba: Satisfactoria

La prueba de aceptación se desarrolló satisfactoriamente, en esta participaron 3 probadores guiados por un especialista del departamento de calidad y en presencia de cliente representado por la jefa del Grupo de aseguramiento de la calidad en el centro. Se encontraron 7 no conformidades en la primera iteración relacionadas con la estructura y ortografía en el complemento, de ellas 3 no proceden porque se referían al uso de términos poco conocidos e interfaces; todos estos diseñados acorde a las características y necesidades del JMeter. En la segunda iteración de las pruebas ya habían sido arregladas las no conformidades que despuntaron, finalizando este proceso satisfactoriamente.

A continuación de muestra un gráfico con los resultados de las no conformidades encontradas durante el proceso de pruebas en el laboratorio del centro:

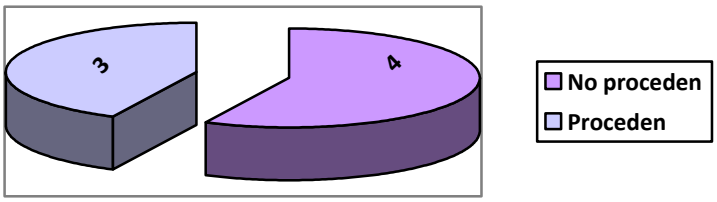

Figura 11.Gráfico resultado de las No Conformidades encontradas durante las pruebas de aceptación.

3.5 Evaluación de la Usabilidad

La usabilidad se define como la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso [43].

En el CEIGE el Grupo de aseguramiento de la calidad define una lista de chequeo, que como objetivo general pretende evaluar las especificaciones del producto. El uso de la lista de chequeo permitirá recoger los puntos eficientes y los ineficientes, definiendo si el indicador a evaluar es crítico o no, utilizando los valores 0 en caso positivo y 1 en caso negativo. El valor final cuantitativo es evaluado por una métrica llamada Grado de usabilidad definida por elementos especificados por el **Estándar ISO 14598.**

Lista de Chequeo Usabilidad ERP-CAL, Cálculo de los resultados arrojados durante la aplicación del chequeo:

El objetivo general de la lista de chequeo permitirá recoger los puntos eficientes y los ineficientes, y con estos evaluar las especificaciones del complemento de configuración. Los resultados de la evaluación serán calculados de la siguiente manera:

Ep=I/N **I**: Cantidad de indicadores evaluados de “0” cuando el elemento revisado no presente errores.

N: Total indicadores de la lista de chequeo.

$U = \frac{\sum_1^k Ep}{k}$ **U**: Evaluación total del sitio.

K: Total de evaluadores que participa en la evaluación.

Ep: Evaluación parcial dada por cada evaluador.

Procedimiento de análisis:

Satisfactoria: cuando U es mayor que 60%.

Aceptable: cuando U está entre 40% y 60%.

Insatisfactoria: cuando U es menor que 40%.

Al finalizar el proceso de evaluación de la usabilidad mediante el cálculo de la métrica se puede realizar una comparación de los resultados obtenido, la herramienta JMeter inicialmente al aplicársele la lista de chequeo obtuvo una evaluación de Aceptable por un total de 0,5433 equivalente al 54% de usabilidad. Seguidamente el uso del complemento de configuración aporto cambios significativos la evaluación arrojada fue de Satisfactoria por un total de 0,9352 equivalente al 93%. A continuación se muestra un gráfico con los resultados de la evaluación por cada uno de los probadores y reflejo de la mejora en los resultados:

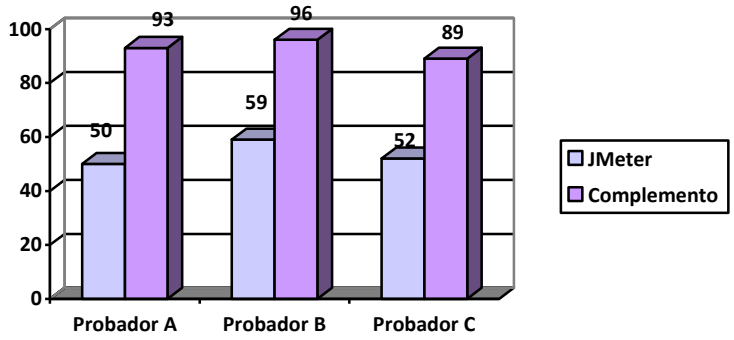

Figura 12.Gráfico de comparación de los resultados de la evaluación de Usabilidad.

3.6 Conclusiones

En el desarrollo de este capítulo se realizó un análisis de la herramienta desarrollada, especificando características de la implementación e interpretando los resultados de las pruebas desarrolladas y una revisión de los Casos de prueba. Combinar la evaluación de la usabilidad y el desarrollo de pruebas según lo definido en la metodología SXP permitió validar el buen funcionamiento y garantía de la solución.

CONCLUSIONES

Luego de concluir el desarrollo de la herramienta para generar el archivo de configuración para el desarrollo de pruebas de carga y estrés utilizando el JMeter se arriba a las siguientes conclusiones:

- Se realizó un estudio de las tecnologías, herramientas y metodologías existentes en el mundo, seleccionando las apropiadas para el desarrollo de la solución.
- Se obtuvo un software de apoyo a la herramienta JMeter logrando una mayor usabilidad de la misma y facilitando una mejor gestión de las pruebas de carga y estrés para aplicaciones de gestión web en el CEIGE.
- Se evaluó la variable usabilidad y se desarrollaron las pruebas de aceptación definidas en la metodología, sumándole a estas la evaluación de integración del complemento con la herramienta JMeter; arrojando resultados satisfactorios para así validar la solución propuesta.

RECOMENDACIONES

- Continuar estudiando las necesidades del centro en relación con los procesos de tesis para incorporarle nuevas funcionalidades.
- Ampliar las funcionalidades del complemento hasta lograr incluir todo tipo de pruebas de rendimiento.
- Se recomienda agregar como funcionalidad exportar a cualquier otro formato elegido por el usuario los resultados de la configuración realizada y los resultados de las pruebas.

REFERENCIAS

[1] Tecnicas_de_evaluacion_de_software- Jurisco Moreno[citado el 19 de enero del 2012]

[2] “UsabilityEngineering” autor: Jakob Nielsen, publicado por: Morgan Kaufmann, SanFrancisco, 1994.ISBN 0125184069[citado el 19 de enero del 2012]

[3] R. S. Pressman. Ingeniería del software. Un enfoque práctico.5taEdición. McGrawHill. [Citado el 20 de enero del 2012]

2011.]http://is.ls.fi.upm.es/docencia/erdsi/Documentacion_Evaluacion_7.pdf[citado el 20 de enero del 2012]

[4] 2011EcuRedhttp://www.ecured.cu/index.php/Pruebas_de_Calidad_de_Software. [Citado el 24 de enero del 2012]

[5] Vegas, Sira, Juristo, Natalia y Moreno, Ana M. 2005. TÉCNICAS DE EVALUACIÓN DE SOFTWARE. [En línea] 17 de octubre de 2005[citado el 25 de enero del 2012]

[6] IEEE 610-1990. Instituto de Ingenieros Eléctricos y Electrónicos. (IEEE Computer Dictionary. Software Engineering Terms. 1990). [citado el 29 de enero del 2012]

[7] SELECTING A DEVELOPMENT APPROACH[citado el 29 de enero del 2012]

[8] Rodolfo Quispe-Otazu. ¿Qué es la Calidad de Software?. Blog de Rodolfo Quispe-Otazu [Internet]. Diciembre 2008. Disponible en: http://www.rodolfoquispe.org/blog/que-es-la-calidad-de-software.php[citado el 29 de enero del 2012]

[9]http://rcci.uci.cu/index.php/rcci/article/view/84.Una experiencia novedosa para el testing desarrollada por un departamento de pruebas de software.Ailyn Febles Estrada, Tayché Capote García, Yeniset León Perdomo, Alionuska Velázquez Cintra, Ramsés Delgado Martínez, Roig Calzadilla Díaz[citado el 29 de enero del 2012]

[10] http://www.mic.gov.cu/sitiomic/servlet/hinfosoc[citado el 30 de enero del 2012]

[11]http://www.combiomed.sld.cu/descargas/generales/COMBIOMED_Profile_Espanol.pdf[citado el 30 de enero del 2012]

[12] Revista Española de Innovación, Calidad e Ingeniería de Software (REICIS), volumen 5, número 2, septiembre 2009.[citado el 1de febrero del 2012]

[13] http://scrum-qa.blogspot.com/2010/09/pruebas-de-stressjmeter.html/ citado 1 de febrero 2012.

[14] http://jakarta.apache.org/jmeter/[citado el 4 de febrero del 2012]

[15] Una explicación de la programación extrema (XP). V Encuentro usuarios xBase 2003 MADRID. Manuel Calero Solís.http://www.apolosoftware.com/[citado el 5 de febrero del 2012]

[16] Robert V. Binder, Testing Object-Oriented Systems: Models, Patterns, and Tools (1999)[citado el 6 de febrero del 2012]

[17] Mi Tecnológico. Temario para las nuevas materias del plan de estudio 2010. http://www.mitecnologico.com/Main/DefinicionCalidadDeSoftware[citado el 6 de febrero del 2012]

[18] Vázquez, Roberto Hugo. Introducción a la calidad de software. [Online] http://gridtics.frm.utn.edu.ar/docs/Introduccion%20a%20la%20Calidad%20de%20Software%20Vazquez.pdf. [Citado el 8 de febrero del 2012]

[19] http://www.calidadysoftware.com/testing/pruebas_funcionales.php/ Ing. Alexander Oré B. [citado el 9 de febrero del 2012]

[20] http://www.librosweb.es/symfony_1_0/capitulo15/automatizacion_de_pruebas.html/ Automatización de pruebas [citado el 9 de febrero del 2012]

[21] Metodología MÉTRICA versión 3.Técnicas y Prácticas.Ministerio de Administraciones Públicas, 2002. [citado el 9 de febrero del 2012]

[22] Mario G. Piattini, Jose A. Calvo-Manzano, Joaquin Cervera Bravo, and Luis Fernandez Sanz. Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software, pages 419-469. Alfaomega, 2004[citado el 9 de febrero del 2012]

[23] Testing en español. Artículos y Herramientas de testing. 2009-2011 http://josepablosarco.wordpress.com/performance-testing/[citado el 10 de febrero del 2012]

[24] Testhouse. Pruebas de Rendimiento. [En línea] 2010. [Citado el: 12 de febrero de 2012.]http://www.es.testhouse.net/pruebas-de-rendimiento

[25] Conferencia de Ingeniería de software. Perfil de calidad. Pruebas y evaluación del software 2008/2009. [citado el 21 de febrero del 2012]

[26] Portal Corporación Cybven C. A. 2011.http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246[citado el 21 de febrero del 2012]

[27] Testing calidad de software. [En línea] [Citado el: 13 de Enero de 2011.]http://www.ivanfl.com/testingcalidad/index.php/pruebas-de-carga/107-jmeter/99-1-primer-contacto.html [citado el 22 de febrero del 2012]

[28] Grupo ARQUISOFT - Johanna Rojas - Emilio Barrios – 2007.http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node68.html[citado el 25 de febrero del 2012]

[29] Peñalver Romero, G.M., “Trabajo de diploma: Metodología ágil para proyectos de software libre”. Ciudad de La Habana, Universidad de las Ciencias Informáticas. 2008. [citado el 25 de febrero del 2012]

[30] http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf/ INSTITUTO NACIONAL DE ESTADISTICA E INFORMATICA - COLECCION CULTURA INFORMATICA/ Econ. Félix Murillo Alfaro[citado el 25 de febrero del 2012]

[31] http://www.ecured.cu/index.php/Rational_Rose_Enterprise_Edition[citado el 27 de febrero del 2012]

[32] http://www.ecured.cu/index.php/Visual_Paradigm[citado el 27 de febrero del 2012]

[33] http://svnbook.red-bean.com/en/1.2/svn-book.pdf/ Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato[citado el 27 de febrero del 2012]

[34] http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf/ Aprenda C++ como si estuviera en primero Escuela Superior de Ingenieros Industriales

de San Sebastián UNIVERSIDAD DE NAVARRA 2008/ Javier García de Jalón,José Ignacio Rodríguez,José María Sarriegui,Alfonso Brazález/editada por Javier García de Jalón. [citado el 27 de febrero del 2012]

[35] I. Jacobson, G. Booch, J. Rumbaugh , "El Proceso Unificado de Desarrollo", Addision Wesley, 2000[citado el 28 de febrero del 2012]

[36] E. Hernández, J. Hernández, C. Lizandra, "C++ Es-tandar", ITP Paraninfo 2001. [citado el 28 de febrero del 2012]

[37] Ruvalcaba, Manuel. 2011. Manuel Ruvalcaba. [En línea] 2011. [Citado el: 3 de marzo del 2012.]http://www.manuelruvalcaba.com/tag/loadrunner/

[38] Oracle Corporation. Información del lanzamiento del IDE NetBeans 6.9.1 [online]. Disponible en http://netbeans.org/community/releases/69/index_es.html. [citado el 6 de marzo 2012]

[39] David Garlan and Mary Shaw. An introduction to software architecture. In Advances in Software Engineering and Knowledge Engineering, volume 1. World Scientific Publishing Co., 1993. . [citado el 10 de abril 2012]

[40] “Arquitectura basada en componentes.” [disponible en:http://www.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes]. [citado el 10 de abril 2012]

[40] “Arquitectura basada en componentes.” [disponible en:http://www.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes]. [citado el 10 de abril 2012]

[41] Diagrama de componentes Ingeniería del Software (3º I.T.I.S., I.T.I.G.) Módulo 2. Tema 12: Modelo de Implementación/disponible en ngeniería+del+Software+(3º+I.T.I.S.%2C+I.T.I.G.)+Módulo+2.+Tema+12:+Modelo+de+Implementación&oq=Ingeniería+del+Software+(3º+I.T.I.S.%2C+I.T.I.G.)+Módulo+2.+Tema+12:+Modelo+de+Implementación&aq=f&aqi=&aql=&gs_l=serp.3...949864.949864.2.952677.1.1.0.0.0.0.0.0..0.0...0.0.kcYCdonBx48&bav=on.2,or.r_gc.r_pw.,cf.osb&fp=e479ee3a3fe03525&biw=1143&bih=680g]

[42]Diagramasde despligue/http://www.ingenieriasoftwaredos.wikispaces.com/Diagramas+de+Artefactos+y+despliegue/ http://www.ecured.cu/index.php/Diagrama_de_despliegue/[citado el 16 de mayo 2012]

[43] Concepto de usabilidad/ISO/IEC 9126/[citado el 25 de mayo 2012]

[44] Application Center Test (ACT) prueba las aplicaciones Web mediante su carga simultánea/

http://support.microsoft.com/kb/307492/es/[citado el 28 de febrero 2012]

[45] Sitio Oficial Bazzar/http://wiki.bazaar.canonical.com/[citado el 20 de febrero 2012]

[46] Sitio oficial Java-Lenguaje de programación/http://www.java.com/es/[citado el 26 de febrero 2012]

[47] Sitio Rational /Características de RUP/http://www.rational.com.ar/herramientas/rup.html/[citado el 26 de febrero 2012]

[48] De la gestión de información a la gestión del conocimiento/http://bvs.sld.cu/revistas/aci/vol14_1_06/aci02106.htm//[citado el 10 de febrero 2012]

[49] El proceso de pruebas en el ciclo de vida/ Dr. Macario Polo Usaola/Departamento de Informática/Paseo de la Universidad/[citado el 20 de febrero 2012]

[50] **Corredor, Germán.** Sun libera su IDE NetBeans para Java y PHP/. http://osum.sun.com/profiles/blogs/sun-libera-su-ide-netbeans-67-1./ [citado el 20 de febrero 2012]

[51]Pruebas de rendimiento y herramientas/ http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246/[citado el 22 de febrero 2012]

[52] Prueba Automática de Carga y Estrés en el Proyecto CICPC/ Liudmila Sánchez Almenares/Universidad /de las Ciencias Informáticas/[citado el 22 de febrero 2012]

[53] Control de versions con Subversion/ Collins-Sussman, Ben; Fitzpatrick, B.W. and Pilato/[citado el 21 de febrero 2012]

[54] Qué es un entorno de desarrollo integrado, IDE/http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/[citado el 21 de febrero 2012]

[55]Lenguajes de programación/ Comparando lenguajes de programación, segunda edición/ Leslie Wilson/[citado el 21 de febrero 2012]

[56] The Art of Soetware Testing, segunda edición / Glenfor F. Myers, Jhon Wiley & Sons Inc. /[citado el 21 de febrero 2012]

BIBLIOGRAFÍA

1. Mantenimiento Avanzado de Sistemas de Información Pruebas del Software. Dr. Macario Polo Usaola

2. Gonzales, J. Las normas de la calidad del software. Addison-Wesley. Iberoamericana. España.

3. Pressman, R. Ingeniería del Software: Un enfoque Práctico 5ta edición. McGraw Hill.

4. An introduction to software architecture. In Advances in Software Engineering and Knowledge Engineering. David Garlan and Mary Shaw.

5. . El Proceso Unificado de Desarrollo. Jacobson, G. Booch, J. Rumbaugh.

6. Trabajo de diploma: Metodología ágil para proyectos de software libre. Peñalver Romero.

7. IEEE. Computer Dictionary. Computer Society.

GLOSARIO

Ajax: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla.

Herramientas CASE: Herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.

HTML (HyperText Markup Language): Lenguaje de Marcado de Hipertexto. Lenguaje en el que se escriben las páginas a las que se accede a través de navegadores WWW. Admite componentes hipertextuales y multimedia.

HTTP: El protocolo de transferencia de hipertexto, usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas web y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

HTTPS: Es la versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información. Es utilizado principalmente por entidades bancarias, tiendas en línea, y cualquier tipo de servicio que requiera el envío de datos personales o contraseñas.

Interfaz: En informática, una interfaz es la parte del programa informático que permite el flujo de información entre varias aplicaciones o entre el propio programa y el usuario. En software también se habla de interfaz gráfica de usuario, que es un método para facilitar la interacción del usuario con el ordenador o la computadora a través de la utilización de un conjunto de imágenes y objetos pictóricos.

Lenguaje: En informática, cuando se habla de lenguaje se refiere generalmente al de programación, conjunto de instrucciones que las aplicaciones necesitan para que el ordenador ejecute determinadas operaciones. Hay lenguaje de alto y bajo nivel, de tercera y cuarta generación, lenguaje natural y lenguaje máquina.

Licencia GPL (GNU Public License): La Fundación para el Software Libre (FSF - Free Software Foundation) está dedicada a eliminar las restricciones de uso, copia, modificación y distribución del software.

Promueve el desarrollo y uso del software libre en todas las áreas de la computación. Específicamente, la Fundación pone a disposición de todo el mundo un completo e integrado sistema de software llamado GNU. La mayor parte de este sistema está ya siendo utilizado y distribuido.

Linux: Linux es una implementación independiente con "espíritu" POSIX (especificación para sistemas operativos). Tiene extensiones System V y BSD, y ha sido escrito completamente a base de aportaciones. Linux no tiene código propietario. Linux está distribuido libremente bajo "GNU Public License". Actualmente solo trabaja en IBM PC (o compatibles) y con arquitecturas ISA e EISA, y requiere un procesador 386 o superior.

Open Source (Código Abierto): Open Source es un término que describe partes de la licencia del movimiento por el software libre.

RUP - Rational Unified Process (Proceso Unificado de desarrollo): Metodología para el desarrollo de Software.

Scripts: Un conjunto de comandos escritos en un lenguaje interpretado.

Servidor: Sistema que proporciona recursos (por ejemplo, servidores de ficheros, servidores de nombres). En Internet este término se utiliza muy a menudo para designar a aquellos sistemas que proporcionan información a los usuarios de la red.

Software libre: Es la denominación del software que brinda libertad a los usuarios sobre un producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

UCI: Universidad de las Ciencias Informáticas.

UML: El Lenguaje Unificado de Modelado (Unified Modeling Language en inglés) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UNIX: Sistema operativo portable, flexible, potente, con entorno programable, multiusuario y multitarea, muy difundido.

XML: Extensible Markup Languaje lenguaje de marcas extensibles.