FACULTAD 15

SUITE DE PRUEBAS PARA EL CENTRO PARA LA INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES

Celia Martínez Rodríguez

Yerandy Martínez Martínez

TUTORES

Ing. SASHA VALDÉS JIMÉNEZ

Ing. GISELLE ALMEIDA GONZÁLEZ

Ciudad de La Habana, 2010

"Año del 52 Aniversario del Triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores de este trabajo y autorizamos a la Facultad 15 de la Universidad de las Ciencias Informáticas; así como al Centro para la Informatización de Gestión de Entidades para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2010.

AUTOR
Celia Martínez Rodríguez

AUTOR
Yerandy Martínez Martínez

TUTOR
ING. Sasha Valdés Jiménez

Co-TUTOR
ING. Giselle Almeida González

RESUMEN

Para asegurar la calidad de un software este debe pasar por un proceso de prueba acorde a sus características, que garantice un rendimiento y funcionalidad óptima. El proceso de pruebas puede convertirse en un proceso difícil si no se tienen los recursos y el personal calificado. Con la aparición de las herramientas automatizadas este trabajo se agiliza considerablemente minimizando los gastos y acelerando las pruebas.

Esta investigación tiene como objeto de estudio la gestión de la calidad en el proceso de desarrollo de software, enmarcando el campo de acción en el conjunto de pruebas que pueden ser automatizadas en el CEIGE y las herramientas disponibles para ello. Una vez detectada la problemática existente se plantea como objetivo general en aras de solucionar la situación, proponer la realización de una serie de pruebas automatizadas así como la utilización de un conjunto de herramientas que optimicen el proceso de pruebas y la calidad del producto final en el CEIGE. Para ello se hizo necesario realizar un estudio de los tipos de pruebas que existen en la actualidad para evaluar un software así como de las tecnologías utilizadas en el centro. Seguidamente se muestran una serie de herramientas que son seleccionadas por sus características y los resultados de las pruebas que se le realizan al sistema.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

Con el avance de las Tecnologías de la Informática y las Comunicaciones (TIC) a escala mundial y el interés de Cuba de insertarse en esta revolución tecnológica, se originan nuevos programas destinados a elevar el nivel cultural de la población y su calidad de vida. En estas circunstancias surge la idea de crear espacios, que además de formar profesionales con un elevado conocimiento sobre el tema, ayuden a potenciar el desarrollo económico y social del país, entre los que se destacan, los Joven Club de Computación y Electrónica y la Universidad de las Ciencias Informáticas (UCI).

Esta última se creó con el objetivo de producir software y servicios informáticos para la sociedad cubana y para el mundo. (1) Es a partir de esta premisa que se han creado un gran número de proyectos, que recogen estudios y desarrollo de aplicaciones que aportan en gran medida a alguna rama de la economía o la sociedad. La aplicación de estos programas de software en diversos procesos empresariales mejora indiscutiblemente las condiciones de trabajo del personal implicado, aumentando la productividad y el rendimiento de la empresa. Todo ello, aparejado a un conjunto de pruebas que se le realizan a los productos de software influye favorablemente en la calidad de los productos y los servicios.

Una de las opciones para facilitar el proceso de prueba, es utilizar las pruebas automatizadas. Las mismas, mejoran la calidad de las aplicaciones, ayudan al equipo de desarrollo de software a investigar los errores, verifican y validan la funcionalidad de los sistemas y aseguran que el producto en desarrollo sea seguro y confiable. Además miden el desempeño del software, mejoran los procesos de gestión del ciclo de vida de los productos y ayudan a equilibrar los procedimientos de pruebas. Gracias a las herramientas de pruebas automatizadas se mejoran los períodos de comercialización porque permiten detectar con eficacia los problemas funcionales, de desempeño y de seguridad generando ahorros en los costos de desarrollo y mantenimiento del software. Por las ventajas que proporciona su uso, aunque no es una obligación, se recomiendan por resultar útil para crear un entorno de desarrollo satisfactorio. (2)

A menudo, a nivel internacional se encuentran problemas con el proceso de pruebas que debe aplicarse durante el desarrollo de los sistemas informáticos. Por lo que la necesidad de mejorarlo es evidente, debido al número de defectos en la entrega, el tiempo y el gasto de recursos (humanos y/o materiales)

durante la realización de las pruebas. Esto impide alcanzar las potencialidades que brindan los sistemas en construcción y los beneficios que aportan.

Para evitar tales inconvenientes en la producción y evaluación del software han surgido herramientas a nivel mundial que realizan diversas pruebas automáticas. Las cuales permiten disminuir el tiempo necesario para llevar a cabo las tareas, reducir la intensidad del trabajo, y lograr que todos los procedimientos se lleven a cabo de manera consistente. En la actualidad, múltiples empresas relacionadas con el mundo de la producción de software, han encontrado en estas herramientas la solución a varios problemas que antes le impedían obtener un resultado satisfactorio. Ejemplo de esto se tiene con el Centro de Ensayos de Software (CES) de la universidad de la República de Montevideo en Uruguay que facilita a sus clientes la utilización de herramientas como las que pertenecen al grupo de herramientas Selenium conformado por Selenium Core, Selenium IDE y Selenium Remote Control, mientras que el Grupo Tecnis, empresa mexicana especializada en soluciones y servicios de software ha lanzado un nuevo servicio de pruebas automatizadas centrado en pruebas de unidad, seguridad, regresión, carga, estrés, entre otras. (3)

En la esfera nacional y muy particular en la UCI, se creó el Centro de Calidad para soluciones Informáticas (Calisoft) que se encarga de verificar la calidad de los productos informáticos que se producen en la universidad. Además se responsabiliza por la evaluación y certificación de productos, procesos y organizaciones según normas nacionales e internacionales y de la asesoría, adiestramiento y formación continua de los especialistas del país en los temas de Calidad e Ingeniería de Software. (4) Dentro de este centro existen áreas que se especializan en temas específicos de calidad, logrando una distribución equitativa al instante de probar un determinado producto.

En el Centro para la Informatización de Gestión de Entidades (CEIGE), se trabaja en la construcción del Sistema Integral de Gestión (CEDRUX), proyecto en el cual se encuentra enfrascada la UCI, resultando necesaria la existencia de un área encargada de garantizar que el software en desarrollo cumpla con los requisitos que define el cliente. Para lograr tal objetivo se traza una estrategia que permite aplicar diferentes tipos de prueba al producto, guiándose por varios estándares previamente definidos, lo que contribuye a realizar una entrega satisfactoria al cliente. Estas pruebas se realizan de forma manual, es decir no existe un apoyo adicional que permita agilizar el proceso. Por lo que sería más factible que se

emplearán herramientas de pruebas automatizadas que mejoren en gran medida la realización del proceso de prueba, como el PHPUnit que realiza pruebas a la base de datos o el JMeter para pruebas de rendimiento, entre otras. Aún así existen deficiencias que dificultan el trabajo en el centro, como lo relacionado con la fuerza de trabajo, conformada principalmente por estudiantes que no se encuentran lo suficientemente capacitados en el tema y no existe plena dedicación a la tarea de pruebas debido a que tanto estudiantes como profesionales necesitan cumplir con otras funciones. Al no aplicar algunas pruebas como las de rendimiento, seguridad, instalación, entre otras, ocasiona que existan vulnerabilidades en el software que no se prueban y por tanto se obtiene un producto ineficiente que no cumple con las expectativas del cliente.

Por lo antes planteado surge el siguiente **problema** a resolver: ¿Cómo mejorar el proceso de prueba en el CEIGE?

El **objeto de estudio** está centrado en la gestión de la calidad en el proceso de desarrollo de software.

El **campo de acción** se enmarca en el conjunto de pruebas que pueden ser automatizadas en el CEIGE y las herramientas disponibles para ello.

Estableciéndose como **idea a defender** que con la utilización de herramientas para la realización de pruebas automatizadas se mejorará el proceso de pruebas en el CEIGE.

El **objetivo general** es proponer la realización de una serie de pruebas automatizadas así como la utilización de un conjunto de herramientas que optimicen el proceso de pruebas y la calidad del producto final en el CEIGE.

Objetivos específicos

- Identificar un conjunto significativo de tipos de pruebas automatizables para el software dentro del proceso de producción definido para el centro.
- Identificar las herramientas que sirvan puntualmente para las pruebas de software identificadas.
- Identificar el momento del ciclo de vida del producto donde deban ser aplicadas las pruebas.

- Validar el uso de las herramientas y la efectividad de las pruebas sobre una aplicación en desarrollo.

Tareas de investigación

- Investigar los tipos de prueba que existen para identificar las automatizables.
- Aplicar una encuesta para conocer qué tipos de prueba se realizan en la UCI y cuáles se realizan en el centro.
- Realizar un estudio de los tipos de herramientas de prueba y sus características para establecer una posición científica y técnica sobre su aplicación.
- Identificar qué tipo de pruebas son necesarias y factibles para el centro, de acuerdo a las características del producto y el proceso de desarrollo.
- Identificar qué herramientas son necesarias y posibles de utilizar en la automatización de las pruebas seleccionadas.
- Planificar pruebas automáticas sobre una muestra de aplicaciones en desarrollo.
- Aplicar las herramientas seleccionadas en los diferentes tipos de pruebas.
- Medir los resultados obtenidos en las pruebas.
- Validar la efectividad de las herramientas seleccionadas y evaluar el resultado de las pruebas.
- Elaborar el informe de la investigación.

Los **métodos científicos** utilizados se describen a continuación:

Métodos históricos: Analizan la trayectoria completa del fenómeno, su condicionamiento a los diferentes períodos de la historia, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales.

Métodos lógicos: Se basan en el estudio histórico del fenómeno, ponen de manifiesto la lógica interna de su desarrollo, de su teoría y haya el conocimiento más profundo de su esencia. Estos métodos expresan

en forma teórica la esencia del objeto, explican la historia de su desarrollo, reproducen el objeto en su forma superior y permiten unir el estudio de la estructura del objeto de investigación con su concepción histórica.

Método dialéctico: Busca las contradicciones existentes y explica los cambios cualitativos que se producen en el sistema y dan paso a un nuevo objeto.

Métodos empíricos: Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional.

Dentro de los métodos empíricos a utilizar se proponen:

- **Método de la observación**

  La observación científica es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado.

- **Métodos particulares**

  Son métodos más específicos que están desarrollados en base a las características propias de cada ciencia y para su aplicación están vinculados a técnicas de recolección de datos característicos de ese tipo de investigación. Dentro de estos se utilizó:

  - **La encuesta**

  Se realiza cuando la información puede ser obtenida a partir de la respuesta que una persona o varias puedan dar a un cuestionario pre elaborado, y las mismas están dispuestas a colaborar con la investigación. La encuesta es semejante a la entrevista pero escrita, donde a través de un conjunto de preguntas se pretende obtener una información sobre el mundo interior del encuestado o su percepción del fenómeno que se investiga, por lo que no puede ser obtenida por observación. Con lo que se pretende analizar el conocimiento que existe sobre el proceso de pruebas de software y las herramientas automatizadas. (5)

El presente trabajo está estructurado en 3 capítulos tal y como se describe a continuación:

Capítulo 1. Fundamentación Teórica: Se analiza estado del arte nacional e internacional y se hace referencia a las herramientas seleccionadas para la creación del trabajo.

Capítulo 2. Propuesta de solución: Se exponen las herramientas escogidas y las razones de su elección para conformar la suite de pruebas de acuerdo a las características del software en desarrollo.

Capítulo 3. Validación de la solución: Se describirá cómo se ha aplicado el proceso y la utilización de las herramientas propuestas en el capítulo 2, se analizarán los resultados obtenidos a partir de las pruebas realizadas con las herramientas propuestas en el entorno de desarrollo así como el resultado del proceso de validación de la propuesta.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN

En el presente capítulo se hace un análisis del estado del arte relacionado al tema de estudio y de los principales conceptos y herramientas que servirán a próximas investigaciones.

1.2. ESTADO DEL ARTE

En la actualidad con la creciente producción de software, la informática se ha vuelto parte indisoluble de nuestro día a día, lo que ha conllevado a la creación incesante de productos que tributen a este sector. Toda esta explosión de programas informáticos ha provocado una difícil competencia y la eficiencia en los sistemas es vital para conseguir rentabilidad en la producción, por eso se ha hecho imprescindible realizarle pruebas a estos sistemas que aseguren su calidad antes de salir al mercado. Inicialmente estas tareas se realizaban manualmente pero debido a la necesidad de acelerar los procesos de prueba se comenzaron a desarrollar herramientas que hicieran estas labores de forma automática o semiautomática lo cual aceleró considerablemente el proceso.

La variedad de las herramientas de pruebas es incalculable, ya que cada una se enfoca a una rama y a un sector específico, esto se debe a que cada empresa o desarrollador independiente las ha creado acorde a sus necesidades. En el caso de las destinadas a las aplicaciones web estas se han orientado a probar la eficiencia de las mismas en cuánto a disponibilidad y seguridad fundamentalmente. Debido a la gran afluencia de usuarios que acceden a ellas y la variedad de los mismos, estos exigen sistemas más rápidos y confiables, mientras que existen otros que tratan constantemente de desmoronarlos para obtener beneficios, ya sea información u otro recurso. Otro elemento a tomar en cuenta en las características de dichas herramientas es su carácter privativo o no, lo cual puede determinar su facilidad de adquisición y uso, ya que muchas veces las privadas exigen elevadas sumas por los permisos de uso y países como Cuba se ven imposibilitados de utilizarlas. Lo que ha propiciado el uso de herramientas libres que en ocasiones no son las óptimas para la solución planteada, y se deban desarrollar estrategias propias.

La calidad del software corresponde uno de los pilares principales para producir y comercializar un producto informático. La producción de software en Cuba debe corresponderse con la velocidad de respuesta y la competitividad que existe en esta rama. Reducir el tiempo de desarrollo, aumentar la

productividad y la satisfacción del cliente, implica acelerar el proceso de evaluación del software. Lo anterior establece utilizar herramientas de automatización de pruebas que han surgido a nivel mundial pero que en Cuba aún no hay suficiente conocimiento sobre el tema, a pesar que existen diversas empresas como: CITMATEL, DESOFT, Segurmática, entre otras que desarrollan sistemas informáticos en varias ramas de la sociedad. Particularmente muy vinculada a la producción del software en Cuba se encuentra la Universidad de las Ciencias Informáticas, que ha logrado realizar grandes aportes de importancia en esta esfera y que además posee un área especializada en el control y seguimiento de la calidad de los sistemas informáticos que produce, estableciendo para ello, guías, estándares, y normas internacionales, como ISO, IEEE, CMMI, y documentación de RUP.
El CEIGE que radica en la propia universidad pretende poner en práctica estas herramientas en el producto que desarrolla, lo que permitirá agilizar el proceso de evaluación disminuyendo tiempo, esfuerzo y gasto de recursos, además que procura aumentar la gama de posibilidades con el objetivo de probar otros aspectos del producto.

1.3. DEFINICIÓN DE CALIDAD DE SOFTWARE

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente"

"El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas". (6)

Partiendo de las definiciones anteriores se puede establecer que *Calidad de Software* es el cumplimiento de los requisitos establecidos para la conclusión de un proyecto de forma satisfactoria, acorde a parámetros y estándares previamente registrados por ambas partes, llevando a término un producto con las cualidades requeridas.

1.4. CONCEPTO DE PRUEBA

Las pruebas son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de

algún aspecto del sistema o componente. Las mismas no pueden asegurar la ausencia de defectos; solo pueden demostrar que existen defectos en el software. Para lograr un mejor entendimiento se proponen los siguientes conceptos de prueba tomados de diferentes fuentes.

La prueba de un sistema se define como el proceso de ejercitar o evaluar el sistema, por medios manuales o automáticos, para verificar que satisface los requerimientos o, para identificar diferencias entre los resultados esperados y los que producen el sistema.

“Las pruebas del software son el proceso de ejecución de un programa con la intención de descubrir un error.” . (7)

En la práctica las pruebas de software son actividades necesarias que se realizan sobre un programa informático para verificar su correcto funcionamiento, así como la existencia de errores de diversas índoles, por lo que sin dudas corresponden a uno de los flujos ingenieriles más importantes en el ciclo de desarrollo de un software.

1.5. OBJETIVO DE LAS PRUEBAS

Estos son algunos de los objetivos que debe perseguir todo diseño según Glen Myers en “The Art of Software Testing” (8):

- 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.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Mientras que Bill Hetzel, en “The Complete Guide To Software Testing”, planteó como objetivos de las pruebas de software (9):

- Planificar las pruebas necesarias en cada iteración.

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

- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados. (7)

Con estos planteamientos se demuestra que el principal objetivos del proceso de prueba es asegurar el correcto funcionamiento del software acorde a los requisitos planteados por el cliente, así como encontrar posibles errores en el sistema de forma eficiente.

1.6. NIVELES DE PRUEBA

En el flujo de trabajo de prueba definido por RUP se buscan errores durante todo el ciclo de desarrollo del software resultando imprescindible conocer los niveles de prueba que existen para la obtención de un software con calidad. Estos especifican qué tipos y métodos de pruebas se deben utilizar en cada uno de ellos y cuáles son sus objetivos.

Unidad

Es una prueba que primero pasa por la revisión del programador, enfocada al código fuente de los componentes y se utiliza para verificar todos los flujos de control. Para llevar a cabo esta prueba se utilizan las técnicas de caja negra y caja blanca de las cuales se aborda a continuación.

- **Prueba de caja negra (Black-box)**

Verifican las especificaciones funcionales y no consideran la estructura interna del programa. Es hecha sin el conocimiento interno del producto. No validan funciones ocultas (por ejemplo funciones implementadas pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados.

En otras palabras, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea los casos de prueba pretenden:

1. Demostrar las funciones del software son operativas.
2. Que las entradas se aceptan de la forma adecuada y que se produce el resultado correcto.
3. Así como la integrada de la información externa (por ejemplo archivos de datos) se mantiene.

- **Prueba de caja blanca (White-box)**

Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código. Las pruebas para la lógica interna tienen que necesariamente emplear la estrategia de caja blanca.

- **Métodos de prueba basados en caja blanca**

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

Mediante estos métodos de prueba de la caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que:

1) Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.

2) Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.

3) Ejecuten todos los bucles en sus límites y con sus límites operacionales.

4) Se ejerciten las estructuras internas de datos para asegurar su validez.

Integración

Los objetivos de la prueba se centran en identificar errores introducidos por la combinación de programas probados unitaria y/o modularmente, determina cómo la base de datos de prueba será cargada, verificar que las interfaces entre las entidades externas (usuarios) y las aplicaciones funcionan correctamente, verificar que las especificaciones de diseño sean alcanzadas y determina el enfoque para avanzar desde un nivel de integración de las componentes al siguiente. Este tipo de prueba maneja dos conceptos fundamentales que a continuación se detallan.

- **Integración descendente (Top-Down)**

  Se integran los módulos moviéndose hacia abajo por la jerarquía de control. Comenzando por el módulo principal, los módulos subordinados se van incorporando a la estructura de una de estas dos formas: primero en profundidad, que integra todos los módulos de un camino de control principal de la estructura, o primero en anchura, que incorpora todos los módulos directamente subordinados a cada nivel, moviéndose por la estructura de forma horizontal.

- **Integración ascendente (Bottom-Up)**

  Empieza la construcción y la prueba con los módulos de los niveles más bajos de la estructura del programa. Dado que los módulos se integran de abajo hacia arriba, el proceso requerido de los módulos subordinados a un nivel dado, siempre están disponibles y se elimina la necesidad de resguardos.

Sistema

Permite asegurar la apropiada navegación dentro del sistema, ingreso de datos, procesamiento y recuperación. Se ejecuta cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. Aceptable para cuando el software se encuentra en la Fase de Construcción.

Aceptación

Prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. A veces se llama “Prueba Piloto”. Se determina que el sistema cumple con lo deseado y se obtiene la conformidad del cliente. Podemos distinguir las:

- **Pruebas alfa:** las realiza el usuario en presencia de personal de desarrollo del proyecto haciendo uso de una máquina preparada para tal fin.
- **Pruebas beta:** las realiza el usuario después de que el equipo de desarrollo les entregue una versión casi definitiva del producto.

1.7. TIPOS DE PRUEBAS

Prueba de especificación

Este tipo de prueba incluye probar la aplicación contra de la documentación que se hizo antes, por ejemplo, que los procesos concuerden con los algoritmos hechos a papel, o que la aplicación tenga todas las funciones que se plantearon.

Prueba de usabilidad

Prueba encaminada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento. Se determina además la calidad de la experiencia de un usuario en la forma en la que éste interactúa con el sistema, se considera la facilidad de uso y el grado de satisfacción del usuario.

Prueba de control de acceso

Se basa en dos aspectos como el nivel de seguridad de la aplicación que verifica que un actor solo pueda acceder a las funciones y datos que su usuario tiene permitido y el nivel de seguridad del sistema que verifica que solo los actores con acceso al sistema y a la aplicación están habilitados para accederla.

Prueba funcional

Prueba centrada en validar las funciones que son objeto de prueba como lo que deben ser, ofreciendo los servicios, métodos o casos de usos requeridos. Deben enfocarse en los requisitos funcionales, las pruebas pueden estar basadas directamente en los Casos de Uso (o funciones de negocio), y las reglas del negocio. Esta prueba es implementada y ejecutada contra diferentes objetos de pruebas, incluyendo unidades, unidades integradas, aplicaciones y sistemas.

Prueba de documentación

Revisiones que se realizan a la documentación a partir de listas de chequeo. Se revisa además la correspondencia y continuidad de los documentos con los generados en etapas precedentes.

Prueba de regresión

Determina si los cambios recientes en una parte de la aplicación tienen efecto adverso en otras partes. Se realizan a partir de la 2da iteración de las pruebas, donde se comprueba que las NC detectadas en la iteración anterior hayan sido respondidas y se prueba nuevamente la aplicación para identificar nuevos errores o defectos introducidos al arreglar los encontrados anteriormente.

Prueba de carga

Usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales.

Prueba de rendimiento ó respuesta

Enfocadas a monitorear el tiempo en el flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccionar los cuellos de botellas y los procesos ineficientes. Compara el rendimiento de un elemento nuevo o desconocido a uno de carga de trabajo de referencia conocido. Determinan los tiempos de respuesta, el espacio que ocupa el módulo en disco o en memoria, el flujo de datos que genera a través de un canal de comunicaciones, entre otros.

Prueba de estrés

Su principal meta es identificar y documentar las condiciones bajo las cuales el sistema falla. Verifica que el sistema funciona apropiadamente y sin errores, bajo condiciones de stress (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible):

- Memoria baja o no disponible en el servidor.
- Máximo número de clientes conectados o simulados (actuales o físicamente posibles)
- Múltiples usuarios desempeñando la misma transacción con los mismos datos.

Prueba de instalación

Enfocada a asegurar la instalación en diferentes configuraciones de hardware y software bajo diferentes condiciones (insuficiente espacio en disco, entre otros.).Además de verificar y validar que el sistema se instala apropiadamente en cada cliente, bajo las siguientes condiciones:

- Instalaciones nuevas, nuevas máquinas a las que nunca se les ha instalado el sistema.
- Actualizar máquinas previamente instaladas con el sistema.
- Instalar versiones viejas en máquinas previamente instaladas con el sistema.

Prueba de recuperación y tolerancia de fallas

Verificar que los procesos de recuperación (manual o automática) restauran apropiadamente la Base de datos, aplicaciones y sistemas, y los llevan a un estado conocido o deseado. Los siguientes tipos de condiciones deben incluirse en la prueba:

- Interrupción de electricidad en el cliente.
- Interrupción de electricidad en el servidor.
- Interrupción en la comunicación hacia el servidor (caídas de red).
- Interrupción en la comunicación con los controladores de disco.

- Ciclos incompletos (procesos de consultas interrumpidos, procesos de sincronización de datos interrumpidos).
- Llaves o apuntadores de base de datos inválidos.
- Elementos corruptos o inválidos en la base de datos.

Prueba de referencia cruzada

Son las pruebas que se realizan a un documento o módulo, a partir de las NC encontradas en otro documento o módulo similar del mismo proyecto o de proyectos relacionados.

Pruebas modulares

Este tipo de pruebas tiene como objetivos:

- Probar la funcionalidad de cada módulo por separado, lo que garantiza detectar y eliminar errores antes de la integración con otros módulos.
- Medir la correspondencia entre lo documentado y las funciones que realmente fueron implementadas en el sistema.

Prueba de liberación

Prueba diseñada e implementada por el Laboratorio de Pruebas de Liberación de la Dirección de Calidad de Software de la Infraestructura Productiva. Se realizan diferentes tipos de prueba en su mayoría de sistema, teniendo en cuenta las características del proyecto que se pruebe. Debe liberarse la última versión del proyecto, a partir de lo pactado con el cliente anteriormente.

Prueba de desarrollador

Prueba diseñada e implementada por el equipo de desarrollo. Estas pruebas pueden realizarse cruzando los programadores, analistas u otro rol, de forma tal que no solo sea la persona la que revise su propio trabajo, pues generalmente una persona ajena es la que más errores puede detectar. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad.

Prueba interna

Prueba diseñada e implementada por el equipo de calidad interna del proyecto. Se pueden realizar pruebas de varios tipos, en dependencia de las características del proyecto en cuestión. Se recomienda que las pruebas se realicen con toda la profundidad requerida, revisando todos los artefactos que sean generados en el desarrollo del producto, para poder verificar que el software se está construyendo correctamente y una vez concluido, validar que el producto desarrollado sea el correcto, teniendo en cuenta los requisitos acordados con el cliente.

Prueba de mutación

Esta prueba está basada en la introducción deliberada de diferentes códigos externos al programa (defectos) para reexaminar si estos defectos pueden ser detectados. Requiere gran disponibilidad de recursos de computación.

Pruebas ergonómicas

Tienen como propósito probar los aspectos ergonómicos del sistema, en otras palabras, las interfaces hombre-máquina en el caso de que éstas existan. Ejemplo: Si los menús son lógicos y legibles, si los mensajes del sistema son visibles, si se puede entender los mensajes de falla, entre otros.

Prueba de configuración

Enfocada a asegurar que funciona en diferentes configuraciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema.

Prueba de contención

Enfocada a la validación de las habilidades del elemento a probar para manejar aceptablemente la demanda de múltiples actores sobre un mismo recurso (registro de recursos, memoria, entre otros.).

Prueba de estructura

Enfocada a la valoración a la adherencia a su diseño y formación. Este tipo de prueba es hecho a las aplicaciones web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.

Prueba de volumen

Prueba centrada en verificar las habilidades de los objetos de prueba para manejar grandes cantidades de datos, tanto en entrada como en salida, o residente en la base de datos. Puede incluir un procedimiento que indique el uso de consultas que devuelvan todo el contenido de la base de datos, o cuando la cantidad de datos de entrada excede a la cantidad establecida de cada campo.

Prueba de seguridad

Prueba centrada en asegurar que los datos o sistemas que son objetos de prueba, son accedidos sólo por los actores que tienen permiso para hacerlo. Esta prueba es implementada y ejecutada contra varios objetos de prueba.

Prueba de validación

Son las pruebas realizadas sobre un software completamente integrado para evaluar el cumplimiento con los requisitos especificados.

Prueba de verificación

Se revisa si el resultado corresponde a la especificación del sistema, es decir, si se está construyendo el sistema de manera correcta.

Prueba de operación

Su objetivo es verificar el sistema en operación por un largo período bajo condiciones normales de uso.

Prueba de réplica

Esta prueba está enfocada a la Base de Datos y a probar que funcione correctamente.

La enorme variedad de pruebas que existen están enfocadas a diferentes niveles de la aplicación y en diferentes momentos de desarrollo del mismo, permitiendo un producto terminado de forma satisfactoria.

Para ello es necesario saber seleccionar de forma acertada las pruebas más factibles para cada entidad, acorde a sus prioridades y objetivos, evitando así realizar trabajos y pérdida de recursos innecesarios.

1.8. AUTOMATIZACIÓN DE PRUEBAS DE SOFTWARE

En nuestros días, que los sistemas son cada vez más grandes y complejos, se hace necesario reforzar el proceso de pruebas. Para ello se aplica el concepto automatización de pruebas, que establece la utilización de herramientas que prueban el sistema sin la interacción de las personas y además permiten una disminución de tiempo, esfuerzo y gasto de recursos. Además que se obtiene una mayor cobertura del software a probar. Todo esto implica lograr una selección factible de acuerdo al ambiente de desarrollo en que se trabaje y a las pruebas de software que se quieran automatizar. Por eso, la selección de la herramienta incorrecta no sólo representa recursos mal gastados, sino que puede crear un obstáculo para el equipo de prueba. Es necesario saber, que la adopción de herramientas para la automatización en una organización de pruebas requiere de un análisis cuidadoso del entorno, y de la situación actual de la organización de prueba en cuestión, desde factores tan elementales como el presupuesto, ya que muchas de ellas tienen precios costosos y por ende no resultarían viables para la empresa. (10)
En esta investigación se coincide en gran medida con otros desarrolladores de software en cuanto a las ventajas de la automatización de pruebas, aunque es necesario destacar que en el caso de la reducción de gastos se debe tener en cuenta que Cuba es un país bloqueado y le es vital poseer una producción altamente rentable, mientras menores sean los gastos de producción más ganancias se obtendrán.

1.9. HERRAMIENTAS DE PRUEBAS AUTOMATIZADAS

SELENIUM IDE

Es un plugin para Firefox, es una pequeña consola que permite grabar movimientos en el navegador, capturar un texto, seleccionar de una lista, marcar una casilla de verificación, para luego reproducirlos tantas veces como se requiera.

Características

- Facilidad de registro y ejecución de los test.
- Autocompletado para todos los comandos.

- Las acciones pueden ser ejecutadas paso a paso.
- Herramientas de depuración y puntos de ruptura (breakpoints).
- Los test pueden ser almacenados como HTML y scripts Ruby, entre otros formatos.
- Soporte para Selenium user-extensions.js.
- Ejecución en varios navegadores.
- Uso de diferentes API's en diferentes lenguajes (PHP, Ruby, JAVA, Javascript, entre otros). (11)

WEBKING DE PARASOFT

Es una herramienta de pruebas web automatizada, que proporciona pruebas exhaustivas y análisis de los sitios y aplicaciones web para asegurar que cumplen con la fiabilidad, seguridad y metas de desarrollo necesarios para sostener su negocio con eficiencia.

Características

- Realización de pruebas complejas integradas en una sola herramienta; análisis de riesgo del sitio web, pruebas funcionales, de carga y desarrollo de pruebas de seguridad y análisis.
- Generación automática de pruebas funcionales para los caminos críticos de su aplicación.
- Diseña automáticamente pruebas de carga realistas, mediante el análisis de sus ficheros de carga del servidor.
- Completa dinámicamente los formularios, con valores de los data source.
- Verifica el cumplimiento de la accesibilidad y estándares de seguridad.
- Se integra con WebSphere Studio Application Developer y Eclipse.
- Análisis de la seguridad integrada y pruebas de intrusión. (12)

OPEN LOAD

Es la primera solución rápida de optimización de rendimiento basada en navegador, fácil de usar, para las pruebas de carga y stress de aplicaciones y sitios web dinámicos. Puede capturar y traducir cualquier acción del usuario de cualquier sitio web o aplicación web. Genera hasta 1000 usuarios simultáneos con el hardware mínimo. (13)

Aunque al igual que el Selenium IDE necesita de la URL del sistema para acceder al mismo y al ser una herramienta que debe ser configurada como localhost y por lo cual se comporta como proxy. Por ser de libre acceso y adaptable a cualquier plataforma, además de brindar al probador amplia información una vez efectuada la prueba se decidió contarlas como candidatas a la suite.

OPENSTA

Es un software distribuido para probar arquitecturas. Su función principal de es la prueba (HTTP y HTTPS) de ejecución y de carga/tensión. Esto te permite crear diferentes patrones de uso en tu sitio web, los que son grabados como scripts para ser fácilmente modificados y tornados dinámicos. Por consiguiente, estos scripts pueden ser ejecutados nuevamente en la arquitectura distribuida de OpenSTA para emular a miles de usuarios virtuales. OpenSTA puede ser utilizado para prueba de regresiones en páginas web. (14)

FUNKLOAD

Es una web funcional y probador de carga, escrito en Python, cuyo principal uso incluye una prueba de funcionamiento de proyectos web y así como pruebas de regresión. También es capaz de realizar pruebas de rendimiento de su aplicación por la carga de la aplicación Web y el seguimiento de sus servidores que ayuda a identificar los cuellos de botella, lo que da un informe detallado de la medición del desempeño. FunkLoad también expone los errores no superficiales en la superficie de prueba, al igual que el volumen de pruebas o ensayos longevidad. (15)

WEBLOAD

Es una herramienta de código abierto para pruebas de carga, pruebas de tensión y pruebas de rendimiento que está patrocinado por RadView. Se puede cargar a la prueba cualquier aplicación de Internet, incluidas las aplicaciones que utilizan la Web 2.0 y AJAX. Este proyecto se basa en el código de doce años de desarrollo invertido en la propiedad anteriormente WebLOAD, la galardonada solución comercial de pruebas de rendimiento para aplicaciones de Internet de Radview Software. WebLOAD es un gran punto de partida para este proyecto, con una comunidad de más de 1.600 clientes y un historial establecido en cuanto al fondo de su robusta funcionalidad, arquitectura eficiente, extensibilidad, el apoyo, la calidad y su facilidad de uso. (16)

DBUNIT

Permite llevar la base de datos a un estado definido antes de cada prueba y que proporciona clases que nos ayudan a realizar pruebas sobre el contenido de una base de datos. Realiza pruebas de unidad, caja blanca e integración. Es una herramienta de código abierto y libre.

Mediante registros de este tipo se pueden generar distintos conjuntos de datos (datasets) en ficheros xml, que pueden servir como semillas para probar el código que accede a la base de datos en situaciones adecuadas para cada prueba.

Permite importar y exportar el contenido de la base de datos a ficheros XML. (17)

CRUISE CONTROL

Es una aplicación de código abierto basado en Java que permite la compilación automática de proyectos Java, utilizando Ant o Maven. Es comúnmente utilizada en integración continua que cada cierto tiempo, o cuando hay cambios en el gestor de versiones (por ejemplo CVS o Subversion), hace una compilación y ejecuta tests (más cualquier otra cosa que esté configurada en Ant o Maven) y una vez acaba presenta el resultado. Esta presentación puede ser en HTML, por correo electrónico, RSS, Jabber, entre otros. (18)

CURL-LOADER

Es una herramienta de código abierto escrito en lenguaje C, simular la carga de aplicaciones y comportamiento de la aplicación de miles y decenas de miles de HTTP / HTTPS y FTP / FTPS clientes, cada uno con su propia fuente dirección IP. La herramienta es útil para la carga de rendimiento de los diversos los servicios de aplicación, para probar los servidores web y ftp y la generación de tráfico. Actividades de cada cliente virtual está conectado y las estadísticas recopiladas incluye información acerca de la resolución, el establecimiento de conexión, el envío de solicitudes, recibir las respuestas, los encabezados y los datos recibidos / enviados, errores de red, TLS / SSL y de aplicación (HTTP, FTP) y los errores de eventos de nivel. (19)

Se puede simular la conexión de más de 100 000 usuarios virtuales al mismo tiempo, todo depende de la cantidad de RAM de la que se disponga, mientras más memoria RAM mayor será la cantidad de usuarios que se pueden simular al mismo tiempo. A cada usuario se le asigna un ip automáticamente después de definir un rango previamente, acepta Ipv4 e Ipv6. Se puede usar por los protocolos HTTP/HTTPS y

FTP/FTPS, telnet, ldap, usando o no TLS/SSL. Muestra un informe sobre cada usuario donde se muestra los tiempos de acceso y de respuesta de la aplicación. (20)

NESSUS

Es un programa que busca cualquier tipo de vulnerabilidad en la red y ofrece un análisis completo de su nivel de seguridad. Consta de dos partes: cliente y servidor; éstas pueden estar instaladas en las misma máquina por simplicidad. Puede grabar informes donde hay enlaces que explican que tipo de vulnerabilidad encontrada, cómo "explotarla" y cómo "evitarla". N*essus* comienza escaneando los puertos con nmap o con su propio escaneador de puertos para buscar puertos abiertos y después intentar varios exploits para atacarlo. Las pruebas de vulnerabilidad, disponibles como una larga lista de plugins, son escritos en **NASL** (Nessus Attack Scripting Language, Lenguaje de Scripting de Ataque Nessus por sus siglas en inglés), un lenguaje scripting optimizado para interacciones personalizadas en redes. Los resultados del escaneo pueden ser exportados en reportes en varios formatos, como texto plano, XML, HTML, y LaTeX. Los resultados también pueden ser guardados en una base de conocimiento para referencia en futuros escaneos de vulnerabilidades. (21)

SHADOW SECURITY SCANNER

Analiza el sistema íntegramente en busca de errores. Es capaz de analizar un rango de IP`s, y luego de hacerlo, brinda un informe en el cual se incluyen los puertos abiertos, agujeros de seguridad, shares compartidos y demás. En cuanto a las vulnerabilidades, produce un informe explicándolas detalladamente, dándoles un nivel determinado de gravedad a cada una, y una posible solución. Incluso posee un algoritmo de seguridad patentado.

Es un completísimo y efectivo escáner de vulnerabilidades para la plataforma de Windows nativa, aunque también examina servidores de cualquier otra plataforma revelando brechas en Unix, Linux, FreeBSD, OpenBSD y Net BSD. También descubre fallos en CISCO, HP y otros equipos de la red. Actualmente, los servicios analizados son: FTP, SSH, Telnet, SMTP, DNS, Finger, HTTP, POP3, IMAP, IRC, Windows Media Service, Terminal Service, NetBIOS, NFS, NNTP, SNMP, Squid, LDAP, HTTPS, SSL, TCP/IP, UDP y servicios del Registro. (22)

PYLOT

Pylot es una herramienta de código abierto para pruebas de rendimiento y escalabilidad de los servicios web. Se ejecuta las pruebas de carga HTTP, que son útiles para la planificación de la capacidad, la evaluación comparativa, el análisis y el sistema de afinación. Genera carga simultánea (HTTP solicitudes), verifica las respuestas del servidor, y produce informes con las métricas. Suites de pruebas se ejecutan y supervisar desde una GUI. (23)

JMETER

JMeter es una herramienta Java desarrollada dentro del proyecto Jakarta, que permite realizar Pruebas de Rendimiento y Pruebas Funcionales sobre Aplicaciones Web. JMeter permite realizar pruebas Web clásicas, pero también permite realizar test de FTP, JDBC, JNDI, LDAP, SOAP/XML-RPC, y WebServices (en Beta). Permite la ejecución de pruebas distribuidas entre distintos ordenadores, para realizar pruebas de rendimiento. Además activar o desactivar una parte del test, lo que es muy útil cuando se está desarrollando un test largo, y se desea deshabilitar ciertas partes iniciales que sean muy pesadas o largas. Tiene la forma de generar un caso de prueba a través de una navegación de usuario.

JMeter como herramienta de prueba dispone de varios componentes que facilitan la elaboración de los escenarios de prueba con la ventaja de simular para cada uno de esos escenarios miles de usuarios. (24)

ZEND STUDIO FOR ECLPISE CON PHPUNIT

PHPUnit es una librería que puede ser usada en la fase de pruebas de aplicaciones PHP. Está hecho para correr pruebas y analizar resultados de manera sencilla. PHPUnit es una migración del popular JUnit utilizado en desarrollos Java, integra casos de prueba basadas en la anotación @grouping, soporta objetos Mock, pruebas de bases de datos, cálculo de métricas de software, documentación ágil, entre muchas otras características. (25) (26)

WAPT PRO

Herramienta para cargar y estresar una aplicación web, de fácil uso, consistente, que te permite analizar el rendimiento y encontrar cuellos de botellas según distintas configuraciones. Ofrece simulaciones precisas de la navegación realizada por un usuario, admite diferentes usuarios en un único test, válido para

aplicaciones dinámicas y contenidos HTTP/SSL y devuelve detallados informes y datos sobre los tests realizados. (25)

DATA GENERATOR FOR POSTGRES

Es una herramienta diseñada fundamentalmente para trabajar sobre gestores de base de datos que soporten postgres, lo cual facilita su uso bajo estas condiciones. Realiza pruebas de unidad, caja blanca y sistema. Brinda la opción de generar datos en XML, Excel, HTML, CSV o SQL y guarda sus formas de generación de datos para uso posterior. Además puede ser utilizado en diversos sistemas operativos como Windows (XP, Windows 7), Linux (Debían, Ubuntu). (26)

WEBINJECT

WebInject es una herramienta gratuita para pruebas automatizadas de aplicaciones web y servicios web. Puede ser utilizado para probar componentes individuales del sistema que tienen interfaces HTTP (JSP, ASP, CGI, PHP, AJAX, Servlets, formularios HTML, XML y servicios web SOAP, REST, entre otros), y puede ser utilizado como un instrumento de prueba para crear un conjunto de [nivel de HTTP] aceptación automatizada funcional, y pruebas de regresión. Un instrumento de prueba le permite ejecutar casos de prueba y recoger muchos informes de sus resultados. WebInject ofrece resultados en tiempo real y también puede ser usado para monitorear los tiempos de respuesta del sistema. Puede ser utilizado como un marco completo de pruebas que está controlada por el WebInject interfaz de usuario (GUI). Opcionalmente, puede ser utilizado como un corredor de prueba independiente (text / aplicación de consola), que pueden ser integrados y marcos de llamada de prueba o aplicaciones. (20)

Las herramientas mostradas se enfocan fundamentalmente sobre las aplicaciones web por lo que serían adaptables al CEIGE de acuerdo a las características que presenta el sistema en desarrollo y acorde a las funcionalidades que ofrecen y al tipo de prueba que realizan. Además se tuvo en cuenta el sistema operativo que soportan, la licencia (libre o propietaria) y el código (abierto o propietario), las características técnicas, precio, URL de descarga y la URL del crack, el serial, y el patch en caso de tenerlo. En su mayoría son adquiridas mediante internet y en otras ocasiones son facilitadas por el centro. No obstante, estas herramientas pueden presentar dificultades de otra índole quedando sujetas a prueba hasta definir su posterior y definitivo uso. Toda esta información ha sido archivada en una ficha técnica para cada herramienta las cuales pueden ser consultadas en Alfresco.

1.10. CONCLUSIONES

En el presente capítulo se han abordado los elementos teóricos sobre los cuáles se apoyará la Suite de pruebas para el CEIGE. El análisis parte de definir inicialmente la gran variedad de tipos de pruebas que existen además de una explicación de la misma para su mejor comprensión. Posteriormente se abordan las ventajas que puede dar la utilización de pruebas automatizadas, además de valorar las facilidades que le reportarían al CEIGE. Las pruebas automatizadas son llevadas a cabo sobre herramientas, de las cuáles se ha hecho una selección acorde a una serie de características, ya que cada una de ellas se desempeña en una dirección determinada. Además se evidencia la importancia que se le atribuye a la automatización de pruebas para lograr un producto informático libre de errores.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

2.1. INTRODUCCIÓN

Con la existencia de un gran número de herramientas de pruebas automáticas se hace necesario investigar cuáles de ellas se ajustan a las características del proyecto CEDRUX. En el presente capítulo se describirán los aspectos que se tuvieron en cuenta para la selección de las herramientas de prueba.

2.2. HERRAMIENTAS DE PRUEBAS EN LA UCI

Con el objetivo de evaluar el nivel de conocimiento que existe en cuánto al proceso de pruebas de software, las herramientas automatizadas que se utilizan en ellas y por la importancia que se le atribuye en el desarrollo de software se realizó una encuesta en las diferentes áreas de la universidad vinculadas tanto a la producción de software como a la evaluación del mismo.

La encuesta estuvo compuesta por 7 preguntas previamente elaboradas (Ver anexo 1). En las dos primeras se aborda el tema de las pruebas de software que se aplican o no en un área determinada. Las restantes cinco preguntas se enfocan a las herramientas de prueba para de esta forma conocer cuáles se utilizan, si se apoya el uso de las mismas y las posibilidades que brinda su uso. Se realizó en dos áreas escogidas de la universidad, Calisoft y la Subdirección de Calidad del CEIGE para un total de 14 encuestados.

Para lograr un mejor análisis de los datos recogidos se decidió utilizar un método para organizar y clasificar la información a través de la distribución de frecuencias las cuáles varían en dependencia del tipo de variable.
Se utilizó el StatGraphics como herramienta estadística para procesar la información recogida en dicha encuesta. A continuación se muestran las gráficas generadas correspondientes a cada una de las preguntas que conforman la encuesta.

En la figura 1 se puede observar que el grado de conocimiento de los encuestados sobre las pruebas de software es de un ciento por ciento.

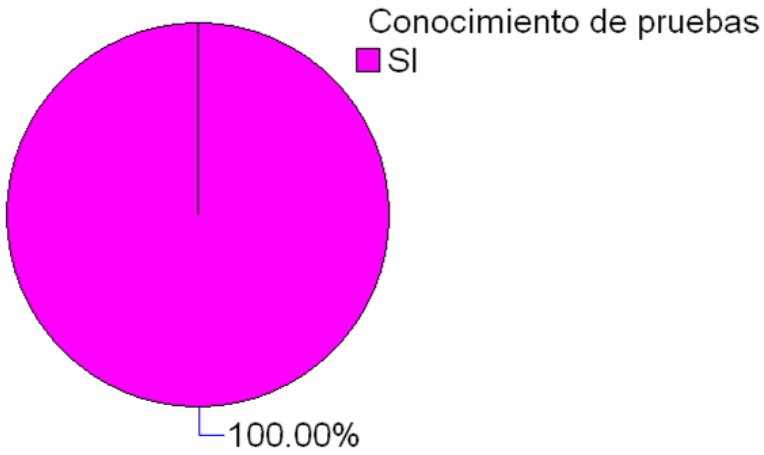


Figura 1 Conocimiento de pruebas de software

En la figura 2 correspondiente a la pregunta 1.1 se específica hasta qué punto el encuestado conoce una determinada prueba, por ejemplo en el caso de la prueba de especificación solo el 42.9 % la conoce y el 57.14 no, además está el caso en que sólo un 7.14 % sabe de otros tipos de pruebas y el 92.86 % no. De manera general las pruebas más conocidas son: caja negra, caja blanca, funcionales, sistema, liberación y documentación.

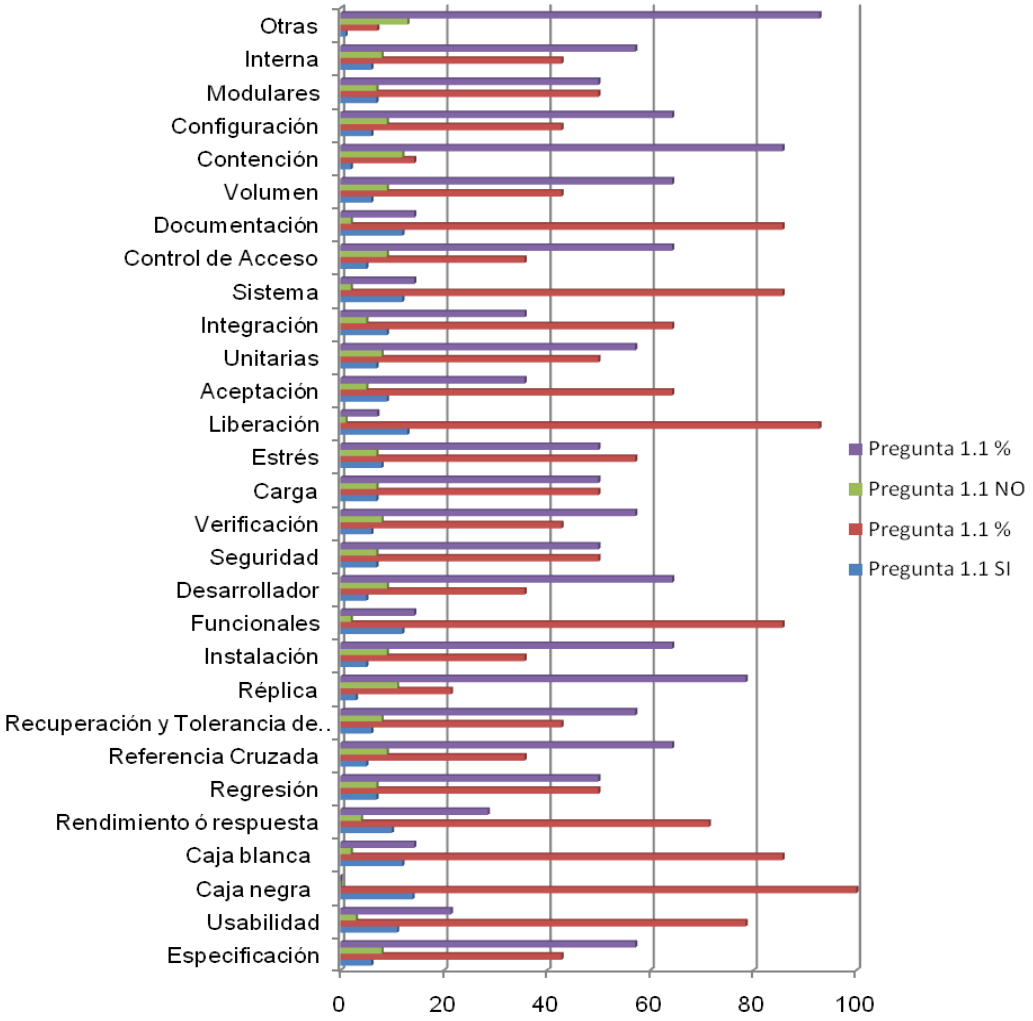


Figura 2 Pruebas específicas de software

En la figura 3 de la pregunta 2 se evidencia un 100 % de utilización de las pruebas de software en cada área donde se realizó la encuesta.

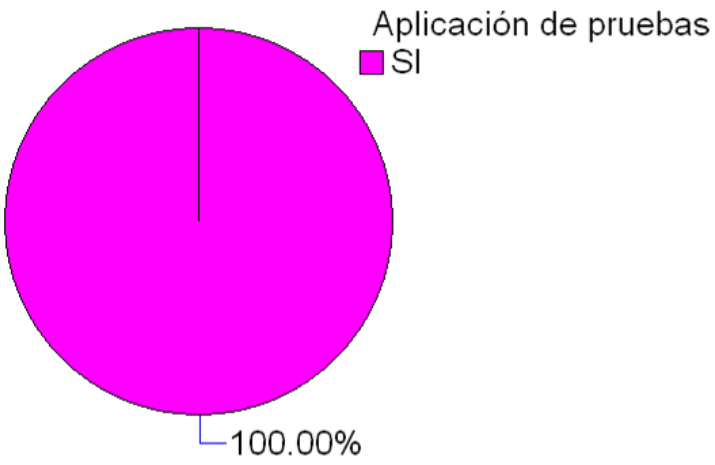


Figura 3 Pruebas en el centro

En la figura 4 de la pregunta 2.1 existen datos críticos en un determinado tipo de prueba como es el caso de referencia cruzada, control de acceso y configuración con un 78.57 %; replica y contención con un 92.86 %, desarrollador y las internas con un 85.71 % lo que demuestra el no uso de este tipo de pruebas en las áreas encuestadas.

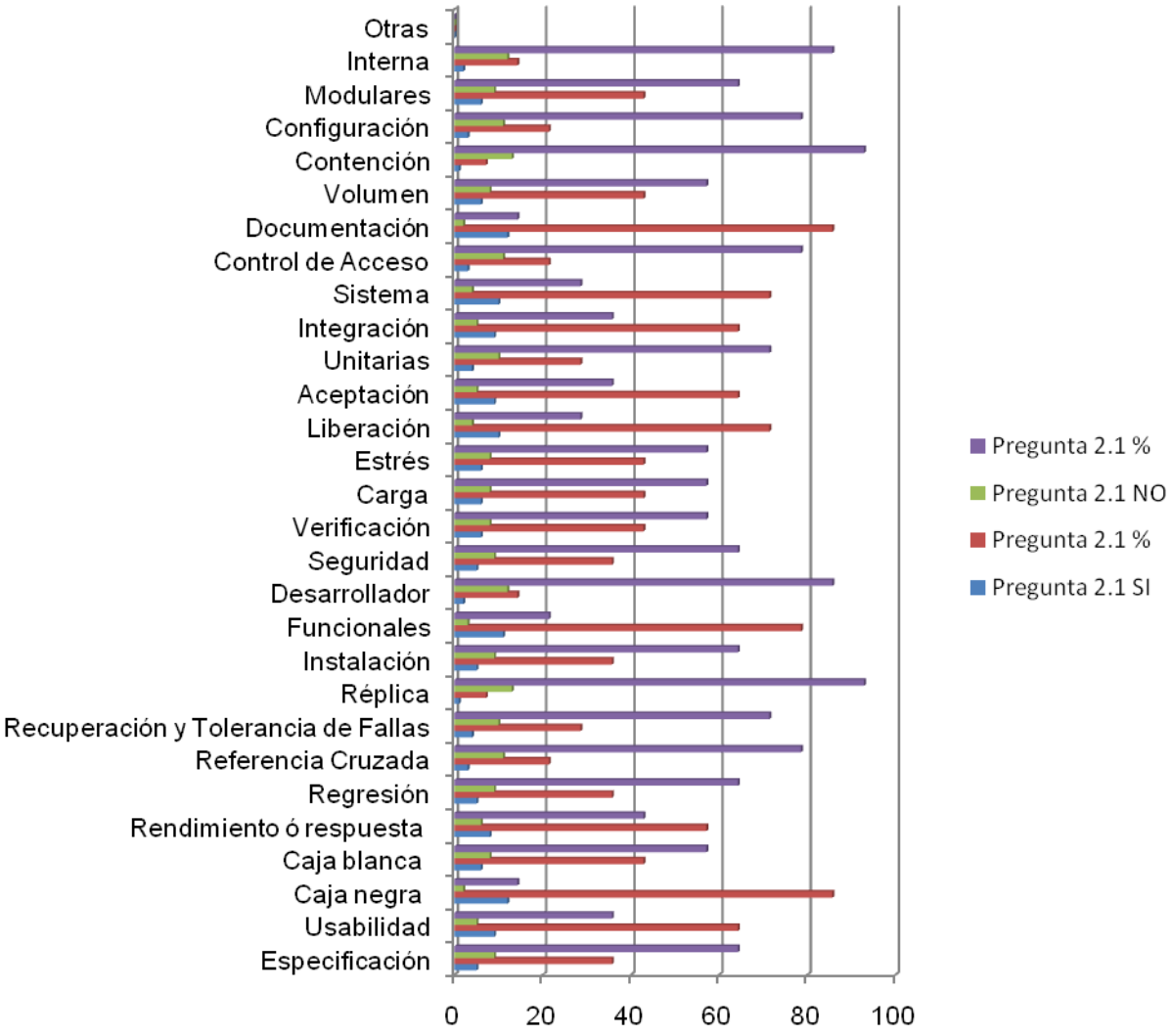


Figura 4 Pruebas realizadas en un área

En la figura 5 de la pregunta 3 se observa cómo el 71.43 % conoce que son las herramientas de prueba y un 28.57% no sabe.

Figura 5 Conocimiento de herramientas de prueba

En la figura a continuación se muestra la tabla de frecuencias generada por el StatGraphics para la pregunta relacionada con el conocimiento de las herramientas.

En la figura 6 de la pregunta 4 se evidencia el grado de aplicabilidad de las herramientas de pruebas en las diferentes áreas para un 57.14 % de uso general por parte de los encuestados.

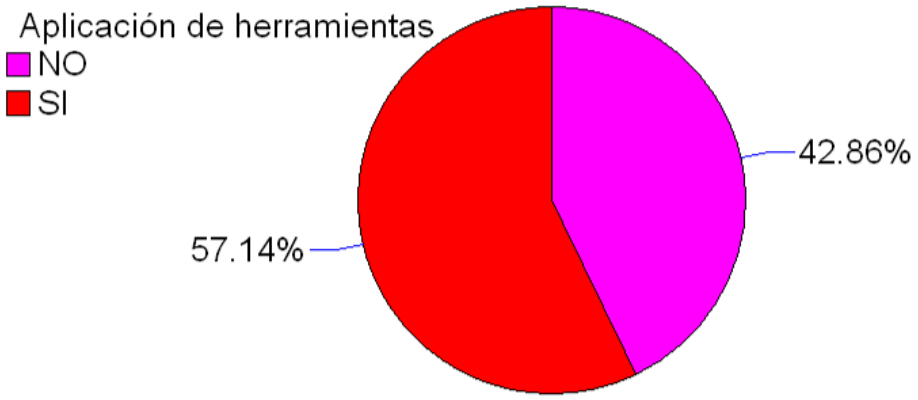


Figura 6 Aplicación en el centro

En la figura 7 correspondiente a la pregunta 5 se observa el dominio que tienen los encuestados en cada una de las herramientas especificadas dándose el caso de un nulo conocimiento sobre el Web developer y el Colorzilla (extensiones de mozilla).

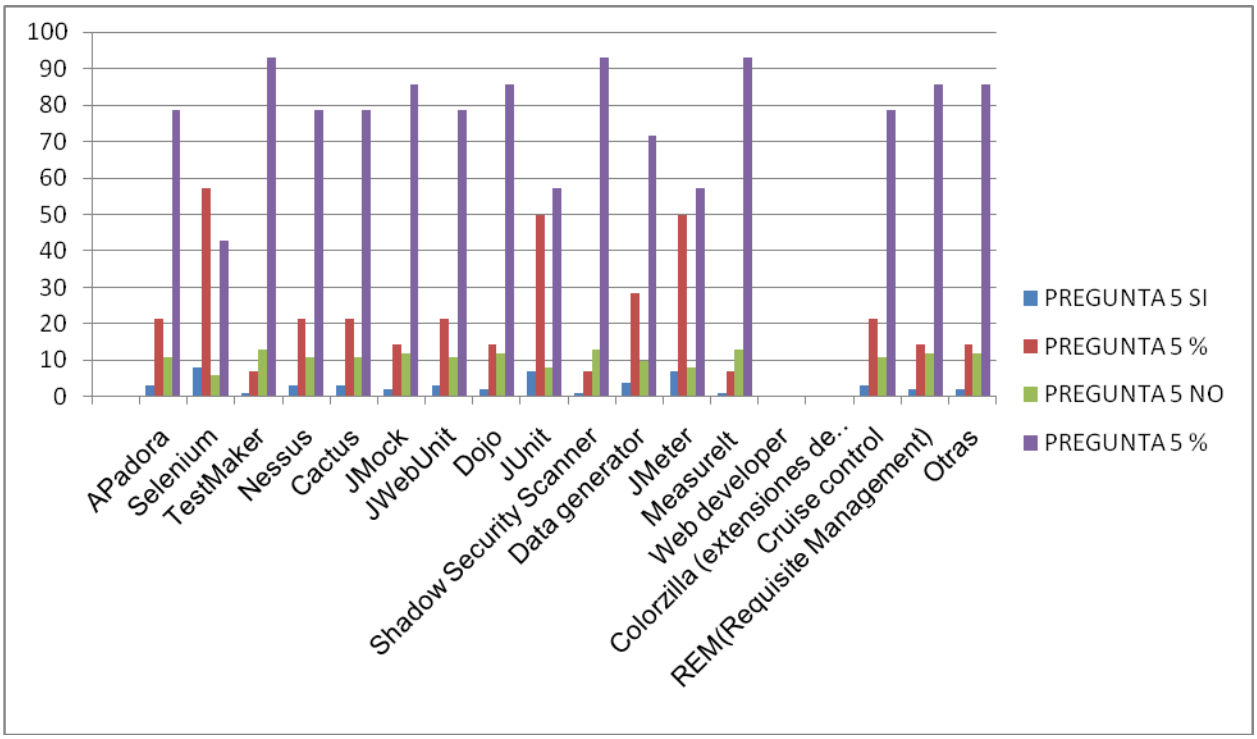


Figura 7 Conocimiento de herramientas de pruebas

La figura 8 de la pregunta 6 muestra el apoyo que existe hacia el uso de las herramientas automatizadas para un 85.71 % de aceptación.

Figura 8 Apoyo al uso de herramientas

La figura 9 de la pregunta 7 muestra la opinión de los encuestados sobre el uso de las pruebas automatizadas de forma general.

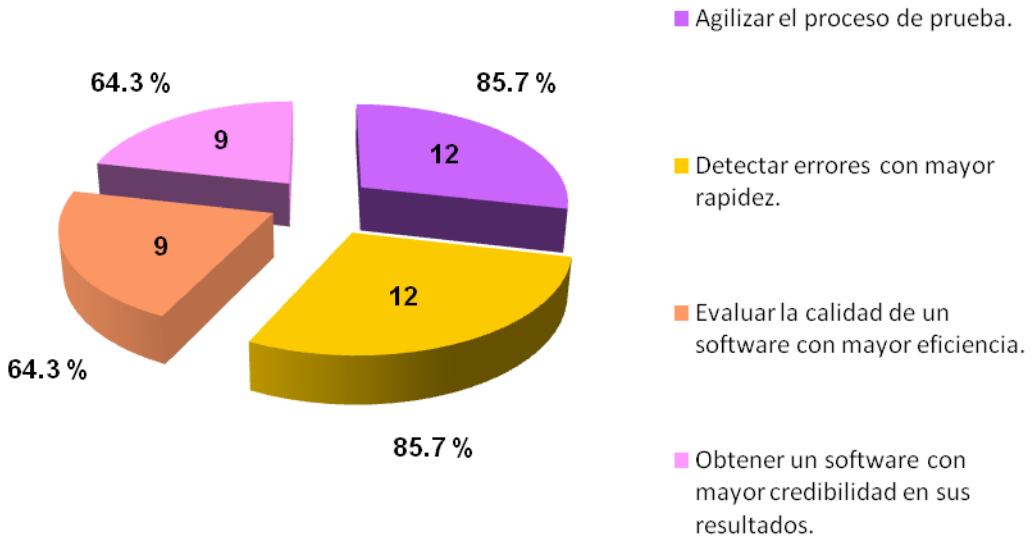


Figura 9 Facilidades de las herramientas de pruebas

2.3. PROCESO DE SELECCIÓN DE LAS HERRAMIENTAS DE PRUEBA

De la encuesta anterior se obtuvo como resultado que a pesar de existir un conocimiento de las pruebas de software, no se ha ampliado el espectro hacia la variedad de las mismas, sus usos y potencialidades, lo que conlleva a desecharlas en muchas ocasiones. Se muestra además como a pesar de estar al tanto del proceso de evaluación de software, algunas de las pruebas que se emplean no se aplican en su totalidad, sólo se hace uso de las más generales dejando a un lado otras también útiles para el sistema. Las herramientas de prueba son desconocidas en casi la mitad de los encuestados por lo que se deduce que no se tenga conocimiento de muchas de las existentes, a pesar de esto se aboga por su uso en gran medida sabiendo que facilitarían exponencialmente el trabajo.

En la selección de las herramientas se ha hecho especial énfasis en las libres y de código abierto por el ahorro económico que representan y las posibilidades de personalización a los intereses tecnológicos del

centro. Además se enfocaron al entorno donde serán utilizadas, y a las características del producto al cual se le aplicarán estas pruebas automatizadas. Una mala elección además de gastar tiempo y recursos, puede incidir en el avance del proceso de prueba y generar un obstáculo difícil de resolver.

2.4. EXPLICACIÓN DE LAS HERRAMIENTAS SELECCIONADAS

OPENLOAD

Resulta útil porque además de ser multiplataforma realiza pruebas de rendimiento al sistema permitiendo así evaluar su potencial. Presenta diversas funcionalidades de fácil ejecución y configuración como:

- Grabar las acciones que hace el probador dentro del sistema.
- El controlador que modela la acción grabada anteriormente.
- El programador que controla las pruebas previamente ejecutadas.
- La sección de análisis donde se obtiene información detallada sobre el sistema a partir de las pruebas ejecutadas.

JMETER

Se caracteriza por sus funcionalidades para realizarle pruebas de estrés, carga y rendimiento a las aplicaciones de manera independiente, al permitir aislar los subsistemas de la aplicación. Además puede ser configurable porque permite definir la cantidad de usuarios que va a simular y las pruebas que se le aplican al marco de trabajo.

De forma general para lograr un mejor entendimiento de su funcionamiento se desglosan a continuación algunas características.

- Se comporta como un proxy local para que a través de él transite toda la información en tiempo real.
- Permite analizar su rendimiento, dadas determinadas condiciones como la navegabilidad en el sistema.

- Numera los errores ocurridos durante la transferencia de datos del servidor al navegador, muchos de estos errores no son visibles para el cliente pero pueden provocar dificultades a corto o largo plazo.
- El muestreo de la información a través de tablas y gráficas facilita la comprensión del flujo de datos y sus características.
- Además brinda información sobre el tiempo de servicio al sistema, el por ciento de rendimiento, el tráfico de información entre el cliente y el servidor tanto los correctos como los incorrectos.

PHPUNIT

La herramienta PHPUnit es un framework que viene agregado al IDE para desarrollar aplicaciones en el lenguaje PHP Zend Studio for Eclipse, realiza pruebas de caja blanca a las aplicaciones web desarrolladas en PHP, vale aclarar que demanda un amplio conocimiento de dicho lenguaje lo que requiere tiempo de preparación para realizar la prueba, ya que todas las pruebas que se le quieran hacer a la aplicación deben ser escritas a código. Una vez concluida la prueba, brinda una serie de gráficas donde se puede verificar la veracidad del método evaluado, el tiempo que demora, por ciento de ejecución del código de cada clase implicada en la prueba, tiempo que demora en ejecutarla y las veces que utiliza un mismo método.

Esta herramienta sin dudas constituye una ayuda para el programador al permitirle probar y corregir cada método al instante de ser elaborado y por ende para el producto Cedrux que desarrolla el CEIGE ya que al ser un proyecto complejo y de gran cantidad de líneas de código reduce el tiempo que se emplea para resolver un determinado error.

DATA GENERATOR FOR POSTGRES

Es una herramienta diseñada fundamentalmente para trabajar sobre gestores de base de datos que soporten postgres. Tiene como fundamental propiedad

- El completamiento de datos de una base de datos de forma automática.
- Simula valores que acceden concurrentemente.

- Valora la velocidad y la carga de información que puede asimilar.
- Comprueba el comportamiento de la base de datos bajo condiciones de estrés y carga de manera eficiente y rápida.

Dadas las características de esta herramienta, como son su compatibilidad con el gestor de base de datos que usa nuestro centro, su facilidad de uso y su alto rendimiento en comparación con otras aplicaciones que realizan funciones semejantes. Vale acotar que el Data Generator for Postgres presenta el inconveniente de no ser libre pero consideramos que dadas sus posibilidades es el más apropiado para realizar las pruebas.

SHADOW SECURITY SCANNER

La herramienta Shadow Security Scanner permite escanear la red en busca de vulnerabilidades lo cual ayuda a evaluar el nivel de seguridad que posee el sistema a través de la red y a su vez evita el surgimiento de algún inconveniente que ponga en riesgo el adecuado funcionamiento del sistema.

2.5. ESCENARIOS DE PRUEBA

PRUEBA DE RENDIMIENTO

Openload

Descripción del escenario para pruebas de rendimiento con Openload	
Precondiciones	• El usuario debe autenticarse en el sistema. • Debe escoger la aplicación a la cual se le realizará la prueba así como su dirección URL. • Definir un nuevo escenario de prueba, ejemplo: Registrar datos. • Definir un nuevo caso de prueba. • Configurar los parámetros que necesita la herramienta

	para ejecutar la prueba. • Salvar la configuración realizada. • Ejecutar la prueba.
Entrada	• Se define la cantidad de usuarios virtuales en esta caso admite hasta 5. • Se configura el tiempo en minutos que demorará la herramienta en realizar la prueba.
Procedimientos	• La herramienta registra cada página del sistema a la cual se accedió durante la grabación así como sus aspectos individuales
Pos condiciones	• La herramienta grabará cada paso que realice el usuario dentro del sistema.
Salida	La herramienta mostrará una serie de datos a través de tablas o gráficos. • Sumario de reportes. • Reporte de usuarios virtuales. • Reporte detallado. • Reporte de errores. • Reporte graficado.
Análisis de resultados	• Permite determinar si los tiempos de respuesta son los esperados bajo determinadas condiciones.

	• Muestra si la aplicación está trabajando con todas sus funcionalidades de forma correcta.

Tabla 1 Openload

Summary Report | Virtual User Report | Detailed Report | Error Report | Graph Report

Test Run	*test Openload*	*CP*
Test Run Start Date:	Thu Mar 11 15:31:46 EST 2010	Thu Mar 11 11:08:33 EST 2010
Test Run Duration (min):	3 min(s) 39 sec(s)	5 min(s) 6 sec(s)
Test Run Status:	completed [Thu Mar 11 15:35:26 EST 2010]	completed [Thu Mar 11 11:13:39 EST 2010]
Business Requirements:		
% Performance	100,0000%	100,0000%
% Availability	100,0000%	100,0000%
% Reliability	96,7320%	97,3901%
Page Time Distribution:		
% Page Time <= 4.0s	100,00%	100,00%
4.0s < % Page Time <= 8.0s	0,00%	0,00%
8.0s < % Page Time <= 12.0s	0,00%	0,00%
% Page Time > 12.0s	0,00%	0,00%
Test Run Statistics:		
Total Virtual Users	5	4
Total Sessions	5	20
Avg Session Time (sec)	199,72	55,78
Total Page Views	30	539
Avg Page Time (sec)	0,55	0,59
Total Requests	153	728
Total Responses	153	728
Hits per sec	0,70	2,38
Returns per sec	0,70	2,38
% 400 Level Errors	3,27%	2,61%
% 500 Level Errors	0,00%	0,00%
% Application Specific Errors	0,00%	0,00%

Figura 10 Sumario

A partir de una prueba realizada a la aplicación con la herramienta de prueba Openload se logró obtener algunos resultados sobre cómo se encuentra funcionando el sistema. En las siguientes gráficas que la propia herramienta ofrece se evidencian diferentes valores asociados a las pruebas **test Openload** y **CP** (Costos y Procesos).Cada gráfica brinda valores de acuerdo a: sumario de reportes, reportes de usuarios virtuales, reporte detallado, reporte de errores y por último el reporte a través de una gráfica. Como se

puede observar en la figura se obtiene diferentes valores, algunos de ellos se marcaran de color rojo debido a que existe algún error, en este caso relacionado con la fiabilidad de la aplicación y con el por ciento de nivel de errores HTTP 400.

Otros de los servicios que ofrece son los relacionado con el reporte de usuarios virtuales, donde se puede observar el tiempo máximo que emplea cada usuario para acceder a una determinada página y en caso de existir algún error se muestra en un reporte más detallado.

JMeter

Descripción del escenario para pruebas de rendimiento con JMeter	
Precondiciones	• Se instala la maquina virtual de java. • Se busca la aplicación a la cual se le van a realizar las pruebas. • Se configura el JMeter como servidor proxy. • Se configura el navegador como: localhost y se arranca el servidor proxy. • Se inicia la navegación por el sitio deseado. • Se registra todo el tráfico desde el servidor hacia el cliente. • Se guarda los resultados de la navegación.
Entrada	• Se seleccionan la cantidad de usuarios a simular • Se específica el tiempo que va a demorar la conexión
Procedimientos	• Se especifican los listeners para registrar los datos acorde a los objetivos. • Se arranca la prueba.
Pos condiciones	• Los listeners registran el mismo tráfico previamente realizado con la cantidad de usuarios definidos.
Salida	• En los listener las tablas y las graficas se muestran el tiempo de respuesta del sistema bajo las condiciones

	configuradas. • Se muestran los componentes enviados desde el servidor así como los errores que se produjeron. • Informa del porciento de rendimiento.
Análisis de resultados	• Permite determinar si los tiempos de respuesta son los esperados bajo determinadas condiciones. • Muestra si la aplicación está trabajando con todas sus funcionalidades de forma correcta. • Los resultados en porcientos muestra si los componentes operan de forma correcta.

Tabla 2 JMeter

Luego de realizar una prueba al sistema con la herramienta se obtiene un informe agregado donde se muestran una serie de datos que exponen el estado en que se encuentra el software con respecto a los módulos probados y se observan una serie de variables que exponen el comportamiento del mismo.

En el listener, se puede observar el instante en el que se van cargando las peticiones http de manera factible o no, el resultado del muestreador, así como la respuesta de sus códigos. Para los casos en los cuales ha sido fallida la petición http se mostrarán en rojo dicha peticiones y para el caso en que sean cargadas satisfactoriamente se mostrarán en verde. No obstante es necesario revisar la respuesta del muestreador ya que en algunas ocasiones se cargan las respuestas a las peticiones en verde y los códigos sin embargo no son los que en realidad tienen que mostrarse.

Data Generator for Postgres

Descripción del escenario para pruebas de carga con Data Generator for Postgres	
Precondiciones	• Se debe tener instalado un cliente local de base de datos o los datos para conectarse remoto (dirección, usuario y contraseña) con la base de datos a probar. • Conectar el Data Generator for Postgres al servidor de base de datos.

Entrada	• Tablas a modificar. • Campos a modificar. • Cantidad de valores a introducir por campos.
Procedimientos	• Definir cómo se van a modificar las tablas. • Especificar el modo de modificar los datos en cuanto incremento y rango. • Especificar si se quiere generar los datos o además crear un archivo SQL con las consultas y así poder utilizarlo en futuras pruebas.
Pos condiciones	• No presenta.
Salida	• Se generan los datos aleatoriamente acorde a los parámetros especificados en la base de datos. • Se genera un archivo SQL con la consulta para posteriores usos.
Análisis de resultados	• Permite comprobar el comportamiento de la base de datos ante un flujo de datos concurrente. • Facilita verificar la eficiencia de la base de datos.

Tabla 3 Data Generator for Postgres

Inicialmente se establece la conexión con el servidor de Dase de Datos introduciendo los parámetros requeridos. Después de acceder satisfactoriamente a la base de datos, el programa muestra todas las tablas existentes en la misma y nos brinda la posibilidad de seleccionar alguna o todas de acuerdo a nuestros objetivos. Posterior a seleccionar las tablas a modificar estas son chequeadas por el programa y muestra la cantidad de campos que poseen así como sus características. Permite modificar el intervalo en que van a variar los valores a introducir, el tamaño de los mismos, así como su orden entre otros aspectos. Se brinda la posibilidad de vaciar las tablas o de mantener los datos anteriores.

Después de seleccionadas las tablas a modificar y el modo en que se va a llevar a cabo, los datos generados pueden ser almacenados para su posterior uso en archivos evitando así repetir el proceso

innecesariamente. Como último paso se ejecuta el llenado de las tablas escogidas y en caso de ser especificado la creación de un script con los datos este es generado al unísono.

Como se muestra esta aplicación es de gran utilidad para el trabajo con las base de datos ya que facilita su completamiento, además que permite valorar el comportamiento de la misma al maniobrar con grandes cantidades de información.

PRUEBA DE SEGURIDAD

Shadow Security Scanner

Descripción del escenario para pruebas de seguridad con Shadow Security Scanner	
Precondiciones	• Se crea una nueva sesión para escanear. • Se escoge el tipo de escáner a ejecutar. • Se llenan los parámetros requeridos para el éxito del escaneo. • Se procede a escanear la red.
Entrada	• Se agregan los host/ip a analizar.
Procedimientos	• La herramienta indica el host, junto con los puertos, auditorías, servicios, recursos compartidos y usuarios.
Pos condiciones	• Describe las características de los puertos abiertos encontrados.
Salida	• Se brinda la información relacionada al host especificado. • Se especifica la gravedad de la

	vulnerabilidad encontrada según los colores : – Verde: Bajo Riesgo. – Celeste: Medio Riesgo. – Rojo: Alto Riesgo.
Análisis de resultados	• Permite determinar la seguridad de la red donde se encuentra instalado el sistema. • Se muestran las irregularidades que puedan existir en la red especificada así como los puertos abiertos que presenta.

Tabla 4 Shadow Security Scanner

Al escanear la red relacionada al número ip de la aplicación se pudo obtener una serie de puertos que utiliza la misma así como la cantidad de paquetes que tiene como se muestra en la siguiente figura.

PRUEBA DE CAJA BLANCA

PHPUnit

Descripción del escenario para pruebas de caja blanca con PHPUnit	
Precondiciones	• Se escoge la clase a probar. • Se presiona el clic derecho sobre la clase para escoger la opción realizar. • Se genera una clase automáticamente que implementa la clase a probar.
Entrada	• Se declaran manualmente las pruebas a realizar.
Procedimientos	• Ejecuta la prueba definida anteriormente. • Comprueba automáticamente que no existen errores en la definición de la

	prueba.
Pos condiciones	• No presenta
Salida	• Devuelve si se ejecutó correctamente la prueba. • Brinda estadísticas a través de gráficos.
Análisis de resultados	• Permite determinar si un método devuelve el resultado esperado. • Muestra hasta qué punto un método es fiable y si presenta algún error que dificulte el correcto funcionamiento del sistema.

Tabla 5 PHPUnit

Perspectiva del Perfil de PHP

La Perspectiva del Perfil de PHP puede ser lanzado automáticamente cuando un perfil de sesión está ejecutándose. Esta perspectiva le permite ver toda la información relevante de su scripts.
Este perfil tiene las siguientes vistas:

1- Perfil de monitoreo

Esta vista muestra una lista de las sesiones pruebas ejecutadas con anterioridad. Expandiendo la lista de una sesión de ejecución, se puede seleccionar la vista de perfil que se quiera visualizar.

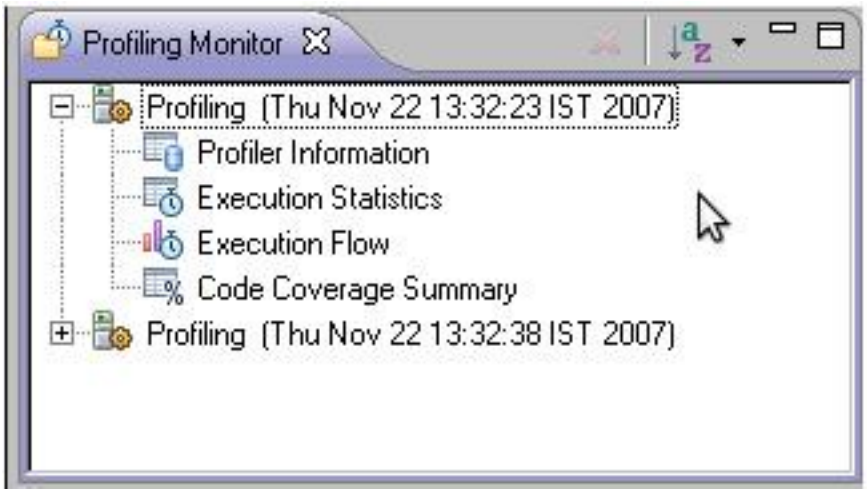


Figura 11 Árbol de componentes

2- Vista de información del perfil

La vista información de perfil muestra información de forma general de la sesión de prueba ejecutada como: duración, fecha de ejecución, números de archivos utilizados para la construcción de la URL solicitada y más. En consecuencia se genera un gráfico de pastel mostrando el tiempo de división de los archivos en la URL.

El lado derecho de la vista muestra el tiempo de división y el lado izquierdo provee la siguiente información:

- URL-La URL analizada.
- Query-Los parámetros específicos de la consulta.
- Path-La localización del primer archivo ejecutado.
- Total Request Time-Tiempo de procesamiento total para la clase completa.
- Number of Files-Número de archivos procesados.
- Date-Fecha y hora en el que el perfil de prueba fue ejecutado.

3-Vista Estadísticas de Ejecución

Esta vista muestra una lista de los archivos que fueron llamados durante la prueba e información detallada del tiempo de ejecución de cada proceso dentro de cada archivo utilizado.

La ventana contiene estadísticas relevantes de cada elemento:

- Funtion-El nombre y localización de cada función.
- Calls Count-Número de veces que fue llamada la función.
- Average Own Time-Promedio de duración sin llamadas internas.
- Own Time(s)-Duración neta de los procesos sin llamadas internas.
- Others Time(s)-Tiempo utilizado en llamar a otros archivos.
- Total Times(s)-Tiempo total del proceso.

Al hacer clic derecho sobre una función de la lista, vemos la opción ´Open Funtion Invocation Stadistics´. Si damos clic en esta opción se abrirá una vista con estadísticas de la función seleccionada, las funciones por las que fue invocada y las funciones que ella misma invocó.

4-Vista Flujo de ejecución

Esta vista muestra el flujo del proceso de ejecución y resume los por cientos y tiempos de ejecución por cada función.

Esta vista muestra la siguiente información por cada función:

- Funtion-Nombre de la función.
- File-El archivo donde se encuentra la función.
- Total Execution time-Por ciento del tiempo utilizado por cada función.
- Duration Time- Tiempo de ejecución de cada función, en segundos.

Clic derecho sobre una función en esta lista nos da las opciones:

- Vista Llamada de Función-Abre la llamada a la función seleccionada en el editor.
- Vista Declaración de Funciones-Abre la declaración de la función seleccionada en el editor.
- Abrir Estadísticas de la Invocación de Funciones-Abre una vista con estadísticas de la función seleccionada, las funciones con las cuáles fue invocada la función seleccionada, y las funciones invocadas por la función seleccionada.

5-Vista Resumen de código Cubierto

Esta vista muestra un resumen de las líneas de código que han sido cubiertas durante el proceso de prueba.

Dicha vista contiene:

- Element-El archivo/proyecto que fue llamado.
- Covered Lines (Visited / Significant / Total)- Muestra el porcentaje de las líneas cubiertas dentro de cada archivo.(Visited= Número de líneas cubiertas/Significant=Número de líneas que fueron significativas o importantes en la llamada a la función /Total=Número total de líneas en el archivo)

Haciendo clic encima de los por cientos de código cubierto, se abrirá un editor que contiene el archivo, con las líneas cubiertas resaltadas.

Como se ha podido constatar luego de un análisis exhaustivo de las herramientas comentadas en el capítulo 1 se realizó una selección de aquellas que realizarán pruebas primordiales a un determinado producto y que cumplieran además con las características del centro donde se desean aplicar y se logró como resultado que Openload, JMeter, Data Generator for Postgres, PHPUnit y Shadow Security Scanner son las más propicias para conformar la suite de prueba.

2.6. CONCLUSIONES

Durante el desarrollo del presente capítulo se analizó el resultado de la encuesta realizada, se proponen herramientas que formarán parte de la suite para facilitar el proceso de prueba de software, propiciando la temprana detección de defectos que a la postre serían difíciles de encontrar y corregir. También se ha descrito el funcionamiento y utilización de las herramientas propuestas, mostrando ejemplos del uso de las mismas en diferentes escenarios elaborados.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN

3.1. INTRODUCCIÓN

En los capítulos anteriores se evidenció con claridad la necesidad de la utilización de herramientas para la realización de pruebas automáticas. En este capítulo se describirá cómo se ha aplicado el proceso y la utilización de las herramientas propuestas en el capítulo 2, las cuáles son el Openload, JMeter, Data Generator for Postgres, PHPUnit y Shadow Security Scanner, el análisis de los resultados obtenidos durante la aplicación de estas herramientas al Centro para la Informatización de Gestión de Entidades así como el resultado del proceso de validación y evaluación de la propuesta.

3.2. VALIDACIÓN DE RESULTADOS

Como vía para validar las herramientas propuestas se decidió planificar un conjunto de pruebas a los diferentes subsistemas de la aplicación con el objetivo de obtener informes detallados sobre su funcionamiento y probar hasta que punto resultan efectivas las herramientas. Dichas pruebas se llevaron a cabo por especialistas de la subdirección de calidad del CEIGE para un total de 9 probadores.

Como se ha explicado dichas herramientas brindan un reporte de la prueba realizada, por lo que a continuación se mostrarán los datos correspondientes a cada herramienta.

JMeter

Los subsistemas elegidos para aplicar las pruebas a través de la herramienta JMeter fueron el portal principal, seguridad, costo y procesos, contabilidad y capital humano. Las pruebas fueron separadas de acuerdo a la cantidad de usuarios virtuales especificados en este caso 50 y 150 respectivamente y para el portal principal de 1000 y 5000.A continuación se muestran una parte de los resultados obtenidos para el caso del portal.

Cada una de estas imágenes ayuda al probador a tener una perspectiva más amplia del funcionamiento del sistema en cuanto a su rendimiento.

sampler_label	aggregate	average	report_medi	report_90%_	report_min	report_max	report_error%	report_rate	report_bandwidth
Grupo de Hilos:Portal	1000	31952	23342	57186	3019	63035	0.395	15.5086849	20.65985516
TOTAL	1000	31952	23342	57186	3019	63035	0.395	15.5086849	20.65985516

Figura 12 Reporte estadístico del JMeter

La siguiente gráfica muestra el tiempo que demora el sistema para responder una determinada acción durante la prueba realizada al subsistema seguridad utilizando la herramienta JMeter.

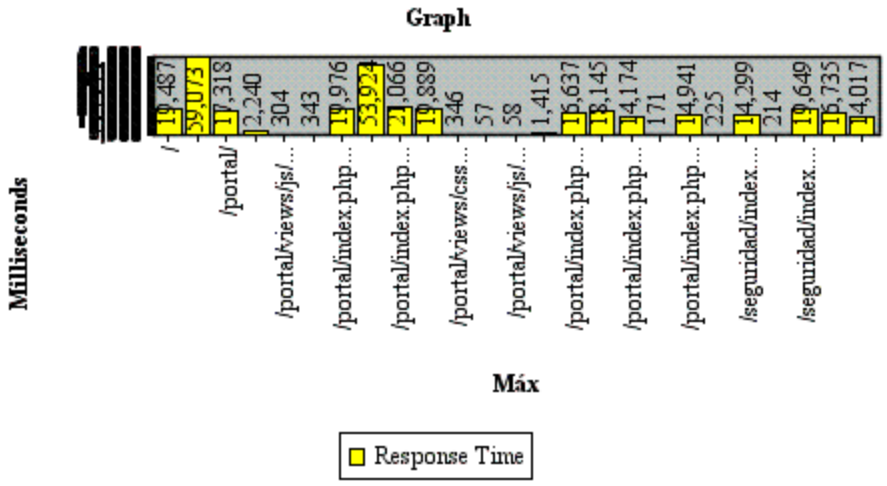


Figura 13 Gráfica del tiempo máximo

Data Generator for Postgres

Esta herramienta tiene la capacidad de generar gran cantidad de datos virtuales que permiten verificar cómo se comporta la base de datos al tener que manipular un alto flujo de información. La base de datos que se tomó como muestra fue la del subsistema Auditoría ya que en el proyecto están separadas por líneas para agilizar el trabajo.

La herramienta muestra el tiempo que toma generar datos para llenar toda la base de datos de la línea Auditoria con 100 elementos por tabla. Esta prueba facilita el trabajo de llenado de la base de datos y permite analizar cómo se comporta ante la confluencia de una gran cantidad de información simultánea.

Otra de las ventajas es la posibilidad de generar un script con una consulta SQL que permite llenar todas las tablas cuantas veces se considere necesario.

Openload

Es una herramienta privada, multiplataforma que realiza pruebas de rendimiento a diferentes aplicaciones web. Para su uso no es necesario un amplio conocimiento de su funcionamiento, aunque si una previa capacitación para lograr obtener resultados satisfactorios. A la hora de efectuar las pruebas sólo se tuvo en cuenta los subsistemas a evaluar Contabilidad y Logística, ya que la propia herramienta brinda una serie de opciones que facilitan el proceso y hacen de lo que antes era algo tedioso un trabajo ameno y rápido. A continuación se muestran ejemplos utilizando dicha herramienta que ameritan su efectividad.

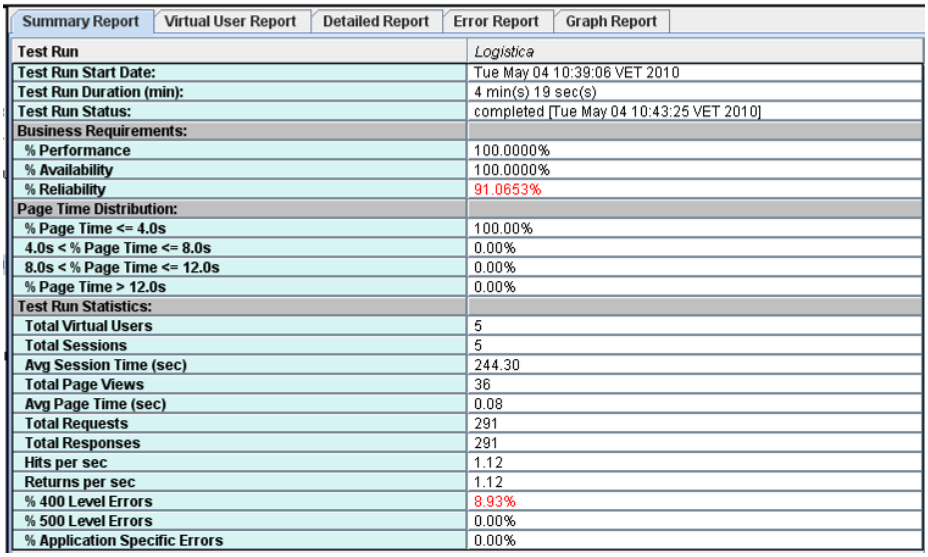
Estas imágenes permiten al probador tener una visión más amplia y exacta del funcionamiento del sistema en cuanto a su rendimiento ante determinada cantidad de usuarios accediendo al mismo y el tiempo que demora en responder una acción realizada.

La figura siguiente muestra un sumario general de la prueba realizada con el Openload. Se puede observar como la tabla se divide en tres sesiones, la primera recoge los requerimientos del sistema: rendimiento, disponibilidad y fiabilidad, la segunda tiene la distribución por tiempo de cada página visitada, y la tercera las estadísticas de la prueba realizada.

Test Run Name:	Contabilidad
Test Run Start Date:	Tue Apr 27 16:07:31 EDT 2010
Test Run Duration (min):	5 min(s) 54 sec(s)
Test Run Status:	completed
Business Requirements:	
% Performance	100.00%
% Availability	100.00%
% Reliability	100.00%
Page Time Distribution:	
% Page Time <= ?s	100.00%
?s < % Page Time <= ?s	0.00%
?s < % Page Time <= ?s	0.00%
% Page Time > ?s	0.00%
Test Run Statistics:	
Total Virtual Users	5
Total Sessions	5
Avg Session Time	5 minute(s) 4 second(s)
Total Page Views	39
Avg Page Time (sec)	0.48
Total Requests	44
Total Responses	44
Hits per sec	0.12
Returns per sec	0.12
% 400 Level Errors	0.00%
% 500 Level Errors	0.00%
% Application Specific Errors	0.00%

Figura 14 Reporte estadístico general

La siguiente imagen muestra los resultados obtenidos utilizando la herramienta Openload en el subsistema Logística se puede ver cómo tanto la fiabilidad como el porciento de errores presentan irregularidades.



Summary Report | Virtual User Report | Detailed Report | Error Report | Graph Report

Test Run	*Logistica*
Test Run Start Date:	Tue May 04 10:39:06 VET 2010
Test Run Duration (min):	4 min(s) 19 sec(s)
Test Run Status:	completed [Tue May 04 10:43:25 VET 2010]
Business Requirements:	
% Performance	100.0000%
% Availability	100.0000%
% Reliability	91.0653%
Page Time Distribution:	
% Page Time <= 4.0s	100.00%
4.0s < % Page Time <= 8.0s	0.00%
8.0s < % Page Time <= 12.0s	0.00%
% Page Time > 12.0s	0.00%
Test Run Statistics:	
Total Virtual Users	5
Total Sessions	5
Avg Session Time (sec)	244.30
Total Page Views	36
Avg Page Time (sec)	0.08
Total Requests	291
Total Responses	291
Hits per sec	1.12
Returns per sec	1.12
% 400 Level Errors	8.93%
% 500 Level Errors	0.00%
% Application Specific Errors	0.00%

Figura 15 Sumario general

PHPUnit

Para realizar las pruebas se utilizaron métodos que tenían un cierto grado de complejidad aceptable y que tuvieran cierta dificultad, con respecto a este elemento recomendamos no utilizar esta herramienta para comprobar algoritmos triviales ya que en vez de ayudar, dificultaría el trabajo de prueba. Después de seleccionados los elementos de pruebas se le deben realizar algunos ajustes para llevar a cabo la prueba y poder generar de forma correcta las gráficas y tablas que muestran los resultados. Es válido que para realizar estas pruebas se necesita conocimiento de php y previa preparación en la herramienta.

En esta imagen se muestra la clase donde se prueban los métodos, los mismos fueron sometidos a prueba con varios juegos de datos para probar su comportamiento con diferentes valores y así obtener un mejor criterio de los resultados.

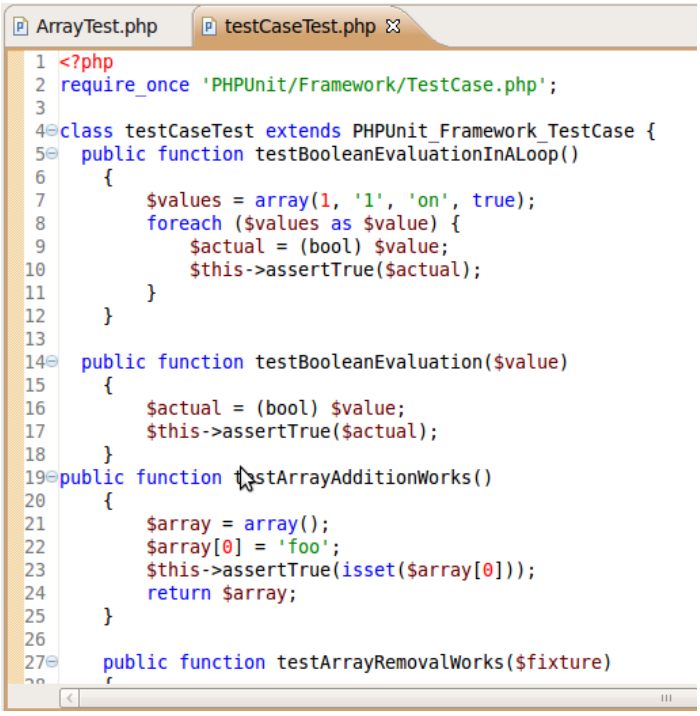
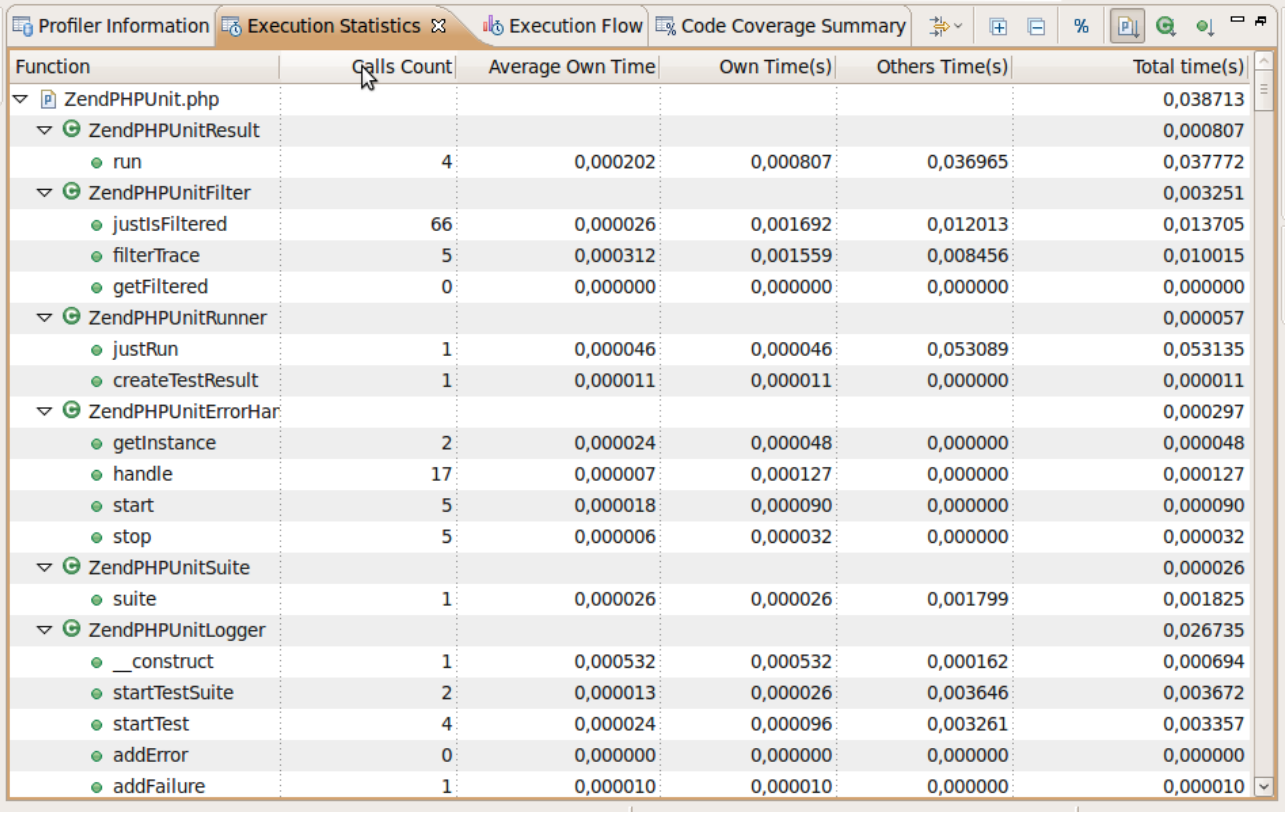


Figura 16 Clase de prueba

Durante la prueba se muestra un perfil de información general que ofrece los tiempos globales del resultado, la distribución de los tiempos durante toda la prueba y el directorio raíz.

La tabla siguiente específica su información en cuanto a los tiempos de ejecución de cada función de forma individual así como el número de veces a los que se le hace llamadas durante la ejecución general de la prueba, lo que facilita conocer el desempeño de cada método por separado y valorar la eficiencia del mismo .



Function	Calls Count	Average Own Time	Own Time(s)	Others Time(s)	Total time(s)
ZendPHPUnit.php					0,038713
ZendPHPUnitResult					0,000807
run	4	0,000202	0,000807	0,036965	0,037772
ZendPHPUnitFilter					0,003251
justIsFiltered	66	0,000026	0,001692	0,012013	0,013705
filterTrace	5	0,000312	0,001559	0,008456	0,010015
getFiltered	0	0,000000	0,000000	0,000000	0,000000
ZendPHPUnitRunner					0,000057
justRun	1	0,000046	0,000046	0,053089	0,053135
createTestResult	1	0,000011	0,000011	0,000000	0,000011
ZendPHPUnitErrorHar					0,000297
getInstance	2	0,000024	0,000048	0,000000	0,000048
handle	17	0,000007	0,000127	0,000000	0,000127
start	5	0,000018	0,000090	0,000000	0,000090
stop	5	0,000006	0,000032	0,000000	0,000032
ZendPHPUnitSuite					0,000026
suite	1	0,000026	0,000026	0,001799	0,001825
ZendPHPUnitLogger					0,026735
__construct	1	0,000532	0,000532	0,000162	0,000694
startTestSuite	2	0,000013	0,000026	0,003646	0,003672
startTest	4	0,000024	0,000096	0,003261	0,003357
addError	0	0,000000	0,000000	0,000000	0,000000
addFailure	1	0,000010	0,000010	0,000000	0,000010

Figura 17 Estadísticas de ejecución

Además ofrece un reporte sobre flujo de trabajo durante la ejecución del método así como los porcientos de los mismos, permitiéndole al programador valorar si los tiempos de ejecución son adecuados y el uso de los métodos es óptimo.

La siguiente tabla muestra los tiempos de la navegación en los métodos dentro de la clase durante todo el tiempo de ejecución así como los porcientos que representan del tiempo total dando un criterio de análisis con respecto al desempeño de cada uno de los mismos.

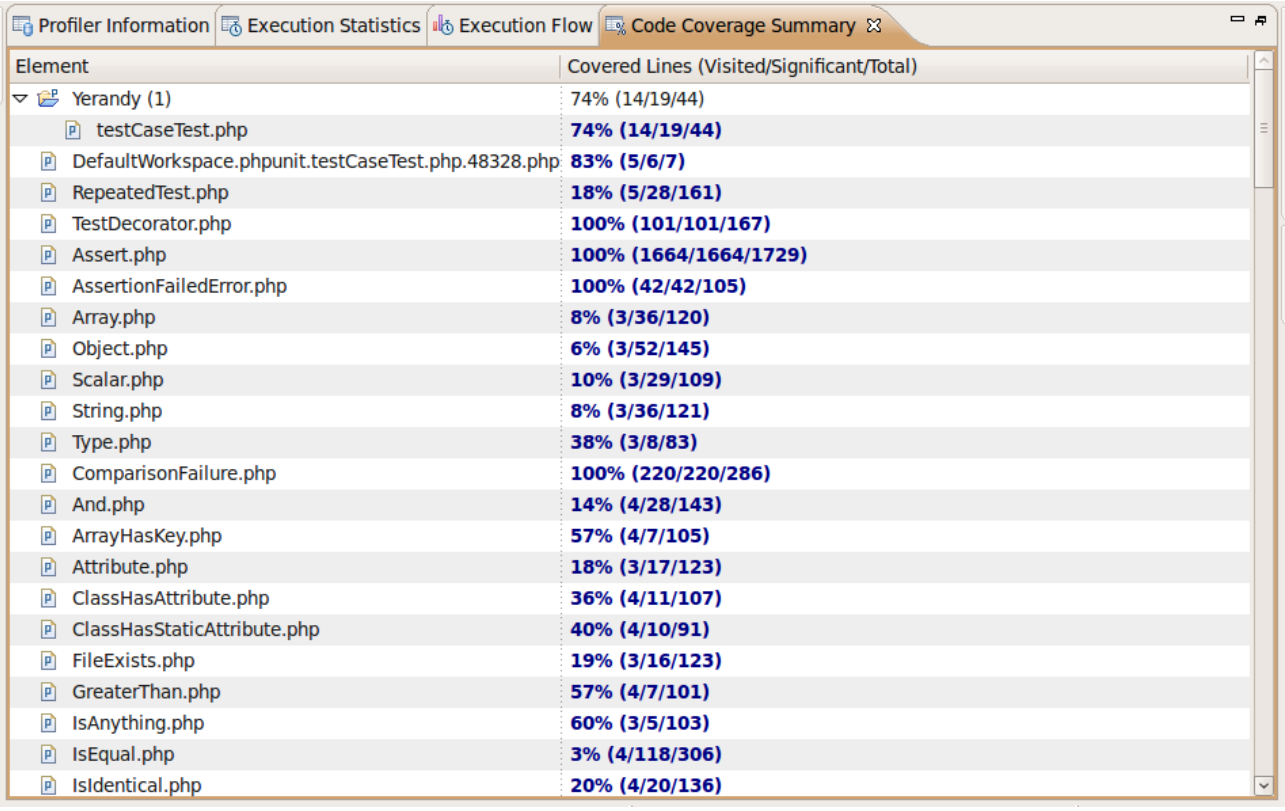


Figura 18 Sumario de porciento del código

Todas estas gráficas permiten dar una valoración detallada del comportamiento de las funciones presentes en la aplicación y si cumplen de forma eficiente su función así como los estándares establecidos además permite optimizar los métodos y modificarlos hasta obtener el resultado deseado.

Shadow Security Scanner

Es un scanner que examina la red en busca de fallos dígase puertos abiertos o agujeros que pongan en peligro el correcto funcionamiento de la red.

Una vez concluida la prueba la herramienta brinda una información general donde se específica la dirección ip y el nombre del host escaneado, tamaño de los paquetes analizados, y la fecha de inicio y fin de la prueba. Además ofrece una visión de los puertos abiertos del ip escaneado y de cada uno de ellos en caso de tenerlo una información adicional en la parte inferior de la interfaz.

3.3. EVALUACIÓN DE TÉCNICA DE LA PROPUESTA DE SOLUCIÓN

Para realizar la evaluación técnica de la propuesta de solución se utiliza el Método de Experto, el cual se basa en la estimación cuantitativa de criterios previamente definidos por parte de expertos en el tema, que permite determinar si se acepta o no la propuesta analizada. A continuación se describen los pasos que se efectuaron para llevar a cabo la evaluación utilizando el Método Experto:

Paso 1: Se elaboran los criterios que fueron utilizados en la evaluación y se agrupan de acuerdo a las características de la propuesta.

Grupo 1: Criterios de Mérito Científico.

1. Valor científico de la propuesta.

2. Contribución científica.

Grupo 2: Criterios de Implantación.

3. Necesidad de empleo de la propuesta.

4. Posibilidades de aplicación.

Grupo 3: Criterios de Flexibilidad.

5. Adaptabilidad a los procesos de prueba.

6. Eficiencia y calidad de los subsistemas evaluados.

7. Capacidad del proceso de evaluación para la admisión de cambios que impliquen mejoras.

Grupo 4: Criterios de Impacto.

8. Impacto en el área para la cual está destinada la propuesta.

9. Optimización del proceso de prueba.

Grupo 5: Criterios de Usabilidad.

10. La propuesta es de fácil entendimiento e instalación.

11. No se requiere conocimientos avanzados por los usuarios.

Paso 2: Se le asigna un peso relativo a cada grupo de criterios de acuerdo al porciento que representa cada grupo del total y los intereses a evaluar.

Grupo 1 → 20 Grupo 2 → 20 Grupo 3 → 30

Grupo 4 → 20 Grupo 5 → 20

Paso 3: Se realiza una selección de nueve expertos en la cual se tiene en cuenta su especialidad, y experiencia en el rol. Además de que se relacione con los temas de Calidad de Software específicamente sobre el proceso de pruebas.

Paso 4: Se hace entrega de la propuesta que se desea validar a todos los expertos para que se documenten sobre el tema de la investigación y luego expresen sus criterios en el modelo. Los expertos conceden pesos de cero a diez a cada factor perteneciente a los criterios establecidos. Además de la apreciación cualitativa con una clasificación final del proyecto en alta, media, baja, y fracaso.

Paso 5: Después de recibir los valores del peso relativo de cada criterio se construye la Tabla 6.

G	C/E	E1	E2	E3	E4	E5	E6	E7	E8	E9	Ep = ΣE / 9
20	C_1	10	9	9	7	7	8	6	8	10	8.22
	C_2	8	8	5	7	9	10	6	6	7	7.33
20	C_3	10	9	7	8	9	8	10	9	7	8.55
	C_4	10	8	10	8	9	10	10	8	9	9.11
30	C_5	10	9	8	8	9	8	9	9	10	8.88
	C_6	10	8	7	9	8	10	8	8	10	8.66
	C_7	10	10	10	7	8	10	9	8	8	8.88
20	C_8	10	9	10	8	10	10	10	9	9	9.44
	C_9	10	9	10	9	8	10	8	7	9	8.88

20	C_{10}	8	10	10	9	9	10	8	10	10	9.33
	C_{11}	10	9	10	6	8	10	8	9	9	8.77
Total		**106**	**98**	**96**	**86**	**94**	**104**	**92**	**91**	**98**	**96**

Tabla 6 Peso otorgado por los expertos a los criterios

Paso 6: Se verifica la consistencia en el trabajo de los expertos, para lo que se utiliza el coeficiente de concordancia de Kendall y el estadígrafo Chi cuadrado (X^2). Siguiendo el procedimiento que se muestra a continuación:

- Sea C el número de criterios que van a evaluarse y E el número de expertos que realizan la evaluación.
- Para cada criterio se determina la ΣE que representa la sumatoria del peso dado por cada experto, Ep que es la puntuación promedio de los pesos correspondientes a cada criterio.
- Se determina la desviación de la media, que posteriormente se eleva al cuadrado para obtener la dispersión S por la expresión: $\mathbf{S = \Sigma(\Sigma E - \Sigma\Sigma E/C)^2}$

Expertos/Criterios	ΣE	ΣE/C	ΣE-ΣΣE/C	$(\Sigma E - \Sigma\Sigma E/C)^2$
C_1	74	6.7272	-4.636	21.492496
C_2	66	6	-12.636	159.668496
C_3	77	7	-1.636	2.676496
C_4	82	7,4545	3.364	11.316496
C_5	80	7,2727	1.364	1.860496
C_6	78	7.0909	-0.636	0.404496
C_7	80	7,2727	1.364	1.860496
C_8	85	7,7272	6.364	40.500496
C_9	80	7,2727	1.364	1.860496

C_{10}	84	7,6363	5.364	28.772496
C_{11}	79	7.1818	0.364	0.132496
ΣΣE/C		78.636		
$S = \Sigma(\Sigma E - \Sigma E/C)^2$				270.545456

Tabla 7 Cálculo de la Dispersión (S) para hallar la concordancia entre los expertos

- Conociendo la dispersión se puede calcular el coeficiente de concordancia de Kendall

  $W = S/ (E^2 (C^3 - C)/12)$

- El coeficiente de concordancia de Kendall permite calcular el Chi cuadrado real:

  $X^2 = E * (C - 1) * W$

Los valores obtenidos se muestran en la Tabla 8.

S	E^2	$C^3 - C$	$E^2 (C^3 - C)$	$E^2 (C^3 - C)/12$	$W=S/(E^2 (C^3 - C)/12)$	$X^2 = E * (C - 1) * W$
270.545456	81	1320	106920	8910	**0.030364**	**2.73276**

Tabla 8 Cálculo de Concordancia

- El Chi cuadrado calculado se compara con el obtenido de la Tabla de Distribución Chi Cuadrado (**Ver Anexo 2**), se toma 1-α=0.99 dónde α es el error permisible, entonces α=0.01, si se cumple que el X^2real < **$X^2(\alpha, c-1)$** se puede decir que existe concordancia en el trabajo de los expertos.

El cálculo arrojó como resultado: **2.73276** < **24.7250**, por lo que se llega a la conclusión de que existe concordancia entre los expertos.

Paso 7: Posteriormente se identifica el peso relativo de cada criterio P y se calcula el Índice de Aceptación (IA) de la propuesta. Para esto se utiliza el procedimiento siguiente:

- Conociendo el número de experto que realizan la evaluación E y la sumatoria de las puntuaciones de cada criterio (C) se puede calcular el peso de cada criterio (P).
- Conociendo el peso de cada criterio P y la cantidad de expertos se puede obtener el valor de P × c, donde (c), es el criterio promedio concebido por los expertos a escala de 5.
- Con el valor anterior se calcula el Índice de Aceptación (IA). $IA = \Sigma(P \times c) / 5$

Criterios	c	P	P x c
C_1	4.11	0.0856	0.3518
C_2	3.67	0.0763	0.2800
C_3	4.28	0.0890	0.3809
C_4	4.56	0.0948	0.4322
C_5	4.44	0.0925	0.4107
C_6	4.33	0.0902	0.3905
C_7	4.44	0.0925	0.4107
C_8	4.72	0.0983	0.4639
C_9	4.44	0.0925	0.4107
C_{10}	4.67	0.0971	0.4534
C_{11}	4.39	0.0913	0.4008
$\sum (P \times c)$			4.3856
IA			0.87712

Tabla 9 Calificación de cada criterio

Paso 8: Por último se determina la probabilidad de existo de la propuesta, ubicando el IA calculado anteriormente en rangos que están predefinidos en la Tabla 9, en dependencia de donde se ubique será la probabilidad de éxito que tenga la propuesta.

0.7 < IA	Alta probabilidad de éxito
0.5 < IA < 0.7	Media probabilidad de éxito
0.3 < IA < 0.5	Baja probabilidad de éxito
IA < 0.3	Fracaso seguro

Tabla 10 Rangos predefinidos de Índice de Aceptación

El IA calculado es **0.87712** lo que significa que existe alta probabilidad de éxito.

Es válido mencionar que las herramientas propuestas se enfocaron en la valoración de los requisitos no funcionales. Luego de realizar las pruebas se demostró con herramientas como el Openload y el JMeter la necesidad de perfeccionar algunos aspectos de disponibilidad de servicios. En muchos casos las ventanas no responden en tiempo de acuerdo a los requisitos, además algunos componentes no

funcionan de forma correcta, que aunque el cliente no lo perciba esto puede provocar fallos o errores en el sistema en determinado momento.

El centro no cuenta con un documento oficial que agrupe los requisitos no funcionales lo que dificulta que exista un estándar para evaluar el funcionamiento del sistema. Algunos de los datos mostrados se obtuvieron mediante la consulta a los profesionales de diferentes áreas autorizados en el tema.

De forma general el sistema cumple en su mayoría las expectativas previstas y se lograron los resultados esperados mostrados en la siguiente tabla.

RESULTADOS ESPERADOS	**RESULTADOS OBTENIDOS**
CUALQUIER INTERFAZ DE LA APLICACIÓN DEBE ESTAR DISPONIBLE EN UN PERIODO DE 0.1 A 0.2 SEGUNDOS.	NO TODAS LAS INTERFACES CUMPLIERON CON EL TIEMPO REQUERIDO.
100 000 USUARIOS CONCURRENTES ACCEDEN A LA INTERFAZ PRINCIPAL DE LA APLICACIÓN, DEBE ESTAR DISPONIBLE EN UN PERÍODO DE 0.5 A 1.0 SEGUNDOS.	NO CUMPLE CON EL REQUERIMIENTO PLANTEADO.
SE INTERCAMBIAN DATOS CON EL SISTEMA, YA SEA MEDIANTE ACCIONES DE INSERCIÓN, BÚSQUEDA, ELIMINACIÓN O MODIFICACIÓN DE DATOS, LA RESPUESTA A LA ACCIÓN REALIZADA DEBE SER EN UN PERÍODO DE 0.1 A 0.7 SEGUNDOS	SE LOGRÓ CUMPLIR CON TIEMPO ESPECIFICADO.
LOS DATOS ACORDES A LOS MÉTODOS.	TODO LOS MÉTODOS FUNCIONARON DE FORMA SATISFACTORIA.

Tabla 11 Análisis de resultados

3.4. CONCLUSIONES

Durante este capítulo se logró conocer más a fondo las características de cada una de las herramientas propuesta para la suite y demostrar su efectividad a partir de las pruebas realizadas con ellas a diferentes subsistemas. El proceso de pruebas permitió validar las fortalezas y debilidades que presentan dichas herramientas dando la posibilidad de una mejor comprensión de estas. Al aplicar las pruebas a estos sistemas se agilizó el proceso vertiginosamente con la calidad requerida demostrando la eficiencia de las herramientas para su uso en el proyecto. En general se realizaron de manera eficiente con una buena organización y todos los procedimientos quedaron plasmados en el presente trabajo, además que se logró la total aceptación de la propuesta por parte de los expertos.

CONCLUSIONES

Después de haber realizado la investigación para solucionar el problema planteado, identificar los objetivos de dicha investigación y dar cumplimiento a las tareas planteadas podemos argumentar la eficiencia de las herramientas de pruebas automatizadas en el centro. Se logró el estudio, selección y prueba de una serie de herramientas para agilizar y humanizar el proceso de pruebas que anteriormente se realizaba de forma manual y era víctima de una serie de dificultades, muchas de las cuáles ahora pueden ser minimizadas. Es válido señalar que las herramientas mencionadas anteriormente no están exentas de inconvenientes pero sus potencialidades las hacen apropiadas para utilizarlas.

Esta investigación permite a cualquier especialista implicado en el proceso de pruebas obtener mayor información acerca de esta nueva tendencia de agilizar la evaluación de un software y de aplicarlo a su entorno de trabajo, logrando obtener mejores resultados en menos tiempo y con menos recursos, ya que se reflejaron las características de cada una de las herramientas estudiadas ahondando más en aquellas que fueron seleccionadas, para las cuales se confeccionó una ficha donde se recogen de forma concisa sus particularidades técnicas. Por lo cual podemos concluir que el uso de herramientas para pruebas automatizadas es una solución factible para el proceso de prueba en el CEIGE.

RECOMENDACIONES

Luego de finalizada la investigación se recomienda lo siguiente:

- Informar a los profesionales y estudiantes del CEIGE de la existencia de herramientas de pruebas automatizadas.
- Exhortar a los implicados en el proceso de prueba a la utilización de estas herramientas.
- Crear un espacio donde puedan ser almacenadas las herramientas propuestas en la investigación.
- Continuar el estudio y aplicación de estas herramientas en el CEIGE.
- Que este trabajo sirva como bibliografía a consultar.

BIBLIOGRAFIA

1. Portal de la Universidad de las Ciencias Informáticas. [En línea] http://www.uci.cu.

2. Technology Evaluation Centers. [En línea] http://test-tools.technologyevaluation.com/es/.

3. Grupo Tecnis. [En línea] http://grupotecnis.blogspot.com/2009/10/grupo-tecnis-lanza-su-servicio-pruebas.html.

4. *InfoCali.* **Tecnológicos, Centro para la Excelencia en el Desarrollo de Proyectos.** 1, Ciudad Habana : s.n., 2010.

5. **Rolando Alfredo Hernández León, Sayda Coello González.** *El paradigma cuantitativo de la investigación científica.* Ciudad Habana : Universitaria, 2002.

6. **Lovelle, Juan Manuel Cueva.** *Calidad de software.* España : s.n., 1999.

7. **Susana González Espinosa, Dania Durán Cutiño.** *Estrategia para la aplicación de pruebas de caja blanca y caja negra al proyeto Registro y Notarías.* Ciudad Habana : UCI, 2008.

8. **Hetzel, Bill.** *The Complete Guide to Software Testing.* 1988.

9. **Myers, Glen.** *The art of software testing.* 1979.

10. **Luis Vinicio León, Abraham Castillo, Elena Ruelas, Aaron.** Software Guru. [En línea] Noviembre-Diciembre de 2006. http://www.sg.com.mx/content/view/347/.

11. **trabajo, Adictos al.** Adictos al trabajo. [En línea] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=seleniumIDE.

12. **Soluctions, Applications LifeCycle.** Applications LifeCycle Soluctions. [En línea] http://www.als-es.com/home.php?location=herramientas/entorno-pruebas/webking.

13. **Opendemand.** Opendemand. [En línea] http://www.opendemand.com.

14. **Osalt.** OpenSTA. [En línea] http://www.osalt.com/es/opensta.

15. **Funkload.** Funkload. [En línea] http://funkload.nuxeo.org.

16. **Webload.** Webload. [En línea] http://www.webload.org.

17. **Sourceforge.** DbUnit. [En línea] http://sourceforge.net.

18. —. Cruise Control. [En línea] http://sourceforge.net.

19. **Wareprise.** Wareprise. [En línea] http://www.wareprise.com/2008/10/29/list-of-top-web-application-load-and-stress-test-tools/es/.

20. **Sourceforge.** Cur-loader. [En línea] http://curl-loader.sourceforge.net/.

21. **Nessus.** Nessus. [En línea] http://www.nessus.org.

22. **Sofonic.** [En línea] http://shadow-security-scanner.softonic.com.

23. **Pylot.** Pylot. [En línea] http://www.pylot.org.

24. **Almenares, Liudmila Sánchez.** *Cómo realizar pruebas de carga y estres en JMeter.* Ciudad Habana : s.n., 2008.

25. **PHPUnit.** PHPUnit. [En línea] http://www.PHPUnit.de.

26. **SGuia.** Wapt. [En línea] http://www.sg.com.mx/guia/node/793.

27.**Iglesias, Mirielys Miranda.** *Elementos de estadísticas para la investigación.* Ciudad Habana : Universidad de la Habana, 2010.

ANEXOS

ANEXO # 1 ENCUESTA REALIZADA A DIFERENTES ÁREAS DE LA UCI Y EL CEIGE

Encuesta para obtener información sobre el conocimiento de las pruebas de software y las herramientas automatizadas para tal fin.

Elaborada por: Celia Martínez Rodríguez

Yerandy Martínez Martínez

Con el objetivo de evaluar el nivel de conocimiento que existe en cuanto al proceso de pruebas de software, las herramientas automatizadas que se utilizan en ellas y por la importancia que tiene dicho proceso en el desarrollo de software; necesitamos su colaboración a través de esta encuesta para recoger su opinión y conocimiento acerca del tema.
De antemano le agradecemos su colaboración y el tiempo que ha dedicado a la misma. Gracias.

1. ¿Conoce usted que es una prueba de software?

   SI___ NO___

   En caso afirmativo, diga:

   1.1. Cuáles de estas pruebas que se exponen a continuación conoce. Marque con una (X).

   ___Prueba de especificación

   ___Prueba de usabilidad

   ___Prueba de caja negra (Black-box)

   ___Prueba de caja blanca (White-box)

   ___Prueba de verificación

   ___Prueba de Carga

   ___Prueba de Estrés

   ___Prueba de Liberación

   ___Prueba de Aceptación

___Prueba de rendimiento ó respuesta (Perfomance)

___Prueba de Regresión

___Prueba de Referencia Cruzada

___Prueba de Recuperación y Tolerancia de Fallas

___Prueba de Réplica

___Prueba de Instalación

___Pruebas Funcionales

___Pruebas de instalación

___Pruebas de desarrollador

___Prueba de seguridad

___Pruebas Unitarias

___Pruebas de Integración

___Prueba de Sistema

___Prueba de Control de Acceso

___Prueba de Documentación

___ Prueba de Volumen

___Prueba de Contención

___Prueba de Configuración

___Pruebas modulares

___Prueba interna

___Otras

¿Cuáles? _____

_____

2. ¿Se realiza en su centro alguna prueba de software?

SI___ NO___

2.1. Cuales considera usted que se realizan en su centro. Marque con una (X).

___Prueba de especificación

___Prueba de usabilidad

___Prueba de caja negra (Black-box)

___Prueba de caja blanca (White-box)

___Prueba de rendimiento ó respuesta (Perfomance)

___Prueba de Regresión

___Prueba de Referencia Cruzada

___Prueba de Recuperación y Tolerancia de Fallas

___Prueba de Réplica

___Prueba de Instalación

___Pruebas Funcionales

___Pruebas de instalación

___Pruebas de desarrollador

___Prueba de seguridad

___Prueba de verificación

___Prueba de Carga

___Prueba de Estrés

___Prueba de Liberación

___Prueba de Aceptación

___Pruebas Unitarias

___Pruebas de Integración

___Prueba de Sistema

___Prueba de Control de Acceso

___Prueba de Documentación

___ Prueba de Volumen

___Prueba de Contención

___Prueba de Configuración

___Pruebas modulares

___Prueba interna

___Otras

¿Cuáles?_____

_____

3. ¿Conoce usted que es una herramienta de prueba?

   SI____ NO____

4. ¿Se utiliza en su centro alguna herramienta de prueba?

   SI____ NO____

5. ¿Conoce algunas de las herramientas de pruebas que se enumeran a continuación? .Marque con una (X).

___ APadora ___ Shadow Security Scanner

___ Selenium ___ Data generator

___ TestMaker ___ JMeter

___ Nessus ___ Measurelt

___ Cactus ___Web developer

___ JMock ___Colorzilla (extensiones de mozilla)

___ JWebUnit ___ Cruise control

___ Dojo ___ REM(Requisite Management)

___ JUnit ___Otras

¿Cuáles? _____

_____

6. Aboga usted por el uso de herramientas de pruebas.

   SI____ NO____

7. Considera usted que aplicar pruebas automatizadas permite:

   a. ___Agilizar el proceso de prueba.
   b. ___Detectar errores con mayor rapidez.
   c. ___Evaluar la calidad de un software con mayor eficiencia.
   d. ___Obtener un software con mayor credibilidad en sus resultados.

ANEXO # 2 TABLA DE DISTRIBUCIÓN CHI CUADRADO

La siguiente tabla es un fragmento de la tabla de Distribución Chi Cuadrado.

p = Probabilidad de encontrar un valor mayor o igual que el Chi cuadrado tabulado.

v = Grados de Libertad.

v/p	0,001	0,0025	0,005	0,01	0,025	0,05	0,1	0,15	0,2	0,25	0,3	0,35	0,4	0,45	0,5
1	10,8274	9,1404	7,8794	6,6349	5,0239	3,8415	2,7055	2,0722	1,6424	1,3233	1,0742	0,8735	0,7083	0,5707	0,4549
2	13,8150	11,9827	10,5965	9,2104	7,3778	5,9915	4,6052	3,7942	3,2189	2,7726	2,4079	2,0996	1,8326	1,5970	1,3863
3	16,2660	14,3202	12,8381	11,3449	9,3484	7,8147	6,2514	5,3170	4,6416	4,1083	3,6649	3,2831	2,9462	2,6430	2,3660
4	18,4662	16,4238	14,8602	13,2767	11,1433	9,4877	7,7794	6,7449	5,9886	5,3853	4,8784	4,4377	4,0446	3,6871	3,3567
5	20,5147	18,3854	16,7496	15,0863	12,8325	11,0705	9,2363	8,1152	7,2893	6,6257	6,0644	5,5731	5,1319	4,7278	4,3515
6	22,4575	20,2491	18,5475	16,8119	14,4494	12,5916	10,6446	9,4461	8,5581	7,8408	7,2311	6,6948	6,2108	5,7652	5,3481
7	24,3213	22,0402	20,2777	18,4753	16,0128	14,0671	12,0170	10,7479	9,8032	9,0371	8,3834	7,8061	7,2832	6,8000	6,3458
8	26,1239	23,7742	21,9549	20,0902	17,5345	15,5073	13,3616	12,0271	11,0301	10,2189	9,5245	8,9094	8,3505	7,8325	7,3441
9	27,8767	25,4625	23,5893	21,6660	19,0228	16,9190	14,6837	13,2880	12,2421	11,3887	10,6564	10,0060	9,4136	8,8632	8,3428
10	29,5879	27,1119	25,1881	23,2093	20,4832	18,3070	15,9872	14,5339	13,4420	12,5489	11,7807	11,0971	10,4732	9,8922	9,3418
11	31,2635	28,7291	26,7569	24,7250	21,9200	19,6752	17,2750	15,7671	14,6314	13,7007	12,8987	12,1836	11,5298	10,9199	10,3410
12	32,9092	30,3182	28,2997	26,2170	23,3367	21,0261	18,5493	16,9893	15,8120	14,8454	14,0111	13,2661	12,5838	11,9463	11,3403
13	34,5274	31,8830	29,8193	27,6882	24,7356	22,3620	19,8119	18,2020	16,9848	15,9839	15,1187	14,3451	13,6356	12,9717	12,3398
14	36,1239	33,4262	31,3194	29,1412	26,1189	23,6848	21,0641	19,4062	18,1508	17,1169	16,2221	15,4209	14,6853	13,9961	13,3393
15	37,6978	34,9494	32,8015	30,5780	27,4884	24,9958	22,3071	20,6030	19,3107	18,2451	17,3217	16,4940	15,7332	15,0197	14,3389
16	39,2518	36,4555	34,2671	31,9999	28,8453	26,2962	23,5418	21,7931	20,4651	19,3689	18,4179	17,5646	16,7795	16,0425	15,3385
17	40,7911	37,9462	35,7184	33,4087	30,1910	27,5871	24,7690	22,9770	21,6146	20,4887	19,5110	18,6330	17,8244	17,0646	16,3382
18	42,3119	39,4220	37,1564	34,8052	31,5264	28,8693	25,9894	24,1555	22,7595	21,6049	20,6014	19,6993	18,8679	18,0860	17,3379
19	43,8194	40,8847	38,5821	36,1908	32,8523	30,1435	27,2036	25,3289	23,9004	22,7178	21,6891	20,7638	19,9102	19,1069	18,3376
20	45,3142	42,3358	39,9969	37,5663	34,1696	31,4104	28,4120	26,4976	25,0375	23,8277	22,7745	21,8265	20,9514	20,1272	19,3374
21	46,7963	43,7749	41,4009	38,9322	35,4789	32,6706	29,6151	27,6620	26,1711	24,9348	23,8578	22,8876	21,9915	21,1470	20,3372
22	48,2676	45,2041	42,7957	40,2894	36,7807	33,9245	30,8133	28,8224	27,3015	26,0393	24,9390	23,9473	23,0307	22,1663	21,3370
23	49,7276	46,6231	44,1814	41,6383	38,0756	35,1725	32,0069	29,9792	28,4288	27,1413	26,0184	25,0055	24,0689	23,1852	22,3369
24	51,1790	48,0336	45,5584	42,9798	39,3641	36,4150	33,1962	31,1325	29,5533	28,2412	27,0960	26,0625	25,1064	24,2037	23,3367
25	52,6187	49,4351	46,9280	44,3140	40,6465	37,6525	34,3816	32,2825	30,6752	29,3388	28,1719	27,1183	26,1430	25,2218	24,3366
26	54,0511	50,8291	48,2898	45,6416	41,9231	38,8851	35,5632	33,4295	31,7946	30,4346	29,2463	28,1730	27,1789	26,2395	25,3365
27	55,4751	52,2152	49,6450	46,9628	43,1945	40,1133	36,7412	34,5736	32,9117	31,5284	30,3193	29,2266	28,2141	27,2569	26,3363
28	56,8918	53,5939	50,9936	48,2782	44,4608	41,3372	37,9159	35,7150	34,0266	32,6205	31,3909	30,2791	29,2486	28,2740	27,3362
29	58,3006	54,9662	52,3355	49,5878	45,7223	42,5569	39,0875	36,8538	35,1394	33,7109	32,4612	31,3308	30,2825	29,2908	28,3361