Testing como Práctica para Evaluar la Eficiencia en Aplicaciones Web

Delvis Echeverría Perez y Ariannis Abella Paumier
Centro Nacional de Calidad de Software
La Habana, Cuba.
{decheverria, abella}@uci.cu

***Resumen*—La investigación está asociada al Testing como práctica para evaluar uno de los atributos de calidad que describe la Norma ISO/IEC 9126 como característica la "Eficiencia", que es según esta norma: La capacidad del producto de software para proporcionar una ejecución o desempeño apropiado, en relación con la cantidad de recursos utilizados usados, bajo condiciones establecidas. Para evaluar esta característica se han identificado un conjunto de tipos de Test, teniendo en cuenta las sub-características de la misma ellas son: Rendimiento, Carga y Estrés, además se propone un procedimiento que guía la gestión y ejecución de los diferentes test. Para las No Conformidades (errores) generados se ha elaborado una clasificación dependiendo de su tipo. Se propone el uso de herramientas automatizadas para la ejecución de los diferentes test, específicamente la herramienta JMETER, usada para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. Se creó una Base de Casos de posibles soluciones que dan respuesta a errores HTTP. Todos estos elementos descritos anteriormente proveen de un marco para la evaluación final de la característica en cuestión en productos de software, proponiéndose un conjunto de métricas que arrojan resultados cuantitativos por cada sub-característica de la Norma o por tipos de test propuestos.**

***Palabras Claves*—Evaluación, Testing, Aplicaciones web.**

I. INTRODUCCIÓN

El desarrollo de productos de software con calidad es un tema primordial y frecuentemente crítico para el éxito de los negocios, la enseñanza, o la seguridad de las personas. Una amplia especificación y evaluación de la calidad de los productos de software es un factor cardinal en el aseguramiento de una calidad adecuada en los productos de software desarrollados. Ello puede ser alcanzado definiendo las características de calidad más apropiadas. La norma ISO/IEC 9126 describe un conjunto de características para la calidad interna y externa, que son además divididas en sub-características que se manifiestan externamente cuando el software se usa como una parte del sistema computarizado, y son un resultado de los atributos internos del software. [1]

Las características y sub-características proveen una consistente terminología sobre la calidad del software y además proveen un marco para especificar los requisitos tanto funcionales como no funcionales de calidad para el software. Esta parte de la ISO/IEC 9126 permite especificar y evaluar la calidad del producto de software desde las perspectivas de aquellos asociados con la adquisición, regulación, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y auditoria del software. Puede ser utilizada por los programadores, los clientes, el personal de aseguramiento de la calidad y los evaluadores independientes, particularmente los responsables de especificar y evaluar la calidad de los productos de software. Categoriza los atributos de calidad del software en seis características, la funcionalidad, la confiabilidad, la usabilidad, la eficiencia, la mantenibilidad y la portabilidad), que a su vez son divididas en sub-características [1].

La presente investigación está asociada a la característica de eficiencia, definida en la ISO 9126 como: La capacidad del producto de software para proporcionar una ejecución o desempeño apropiado, en relación con la cantidad de recursos utilizados usados, bajo condiciones establecidas.

II. TESTING

Las pruebas de software también llamadas "testing", su primera referencia puede ser rastreada a 1950, pero fue solo en 1957 que fue distinguida del debugging [2].

Dijkstra en 1970 presentaba una importante limitación, que "la prueba de software puede ser usada para mostrar la presencia de bugs, pero nunca su ausencia" [3].

Según la IEEE, lo define como una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

El testing es un elemento crítico e imprescindible para la garantía de la calidad, y de ahí la necesidad de aplicarlo. A través del mismo se ve la medida en que las funcionalidades del software se corresponden con las especificaciones establecidas y los datos que va arrojando constituyen un indicador de la fiabilidad del mismo.

Esta propuesta se enmarca en la evaluación de la característica eficiencia de la ISO/IEC 9126.

Dentro de la característica de eficiencia se han identificado un conjunto de tipos de test identificados dentro de las sub-características de la norma:

- Rendimiento
  Enfocadas a monitorear el tiempo en flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccionar los cuellos de botellas y los procesos ineficientes.
- Carga
  Enfocada en 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.
- Estrés
  Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos

compartidos no disponible).

Estos tipos de testing permiten conocer y mitigar los riesgos relacionados con el mal desempeño de las aplicaciones en los entornos de producción y realizar las correcciones necesarias antes de salir al mercado. Se cuantifica la capacidad de la infraestructura, se validan los requerimientos de performance, la escalabilidad de las plataformas y del sistema a probar. De esta manera, la empresa puede conocer qué cantidad de clientes simultáneos soporta su producto, con tiempos y datos razonables sobre la infraestructura y las plataformas propuestas. Asimismo, puede saber si es suficiente el hardware para soportar el nivel propuesto de transacciones y qué expectativa de crecimiento soporta. Es relevante la selección de la herramienta adecuada, la definición de los escenarios de prueba y la configuración del entorno. Una vez ejecutadas las pruebas, es importante la evaluación de resultados para identificar los posibles problemas en la performance actual y las posibilidades de mejora sobre el desempeño del sistema [4].

III. Procedimiento Propuesto

El procedimiento propuesto para la ejecución de los tipos de testing identificados dentro de la característica de eficiencia de describe las siguientes actividades:

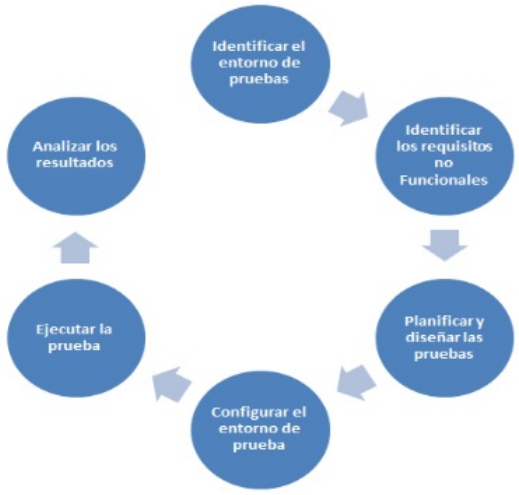

Fig. 1. Actividades del Procedimiento Propuesto.

- Identificar el entorno de pruebas:
  Identificar el entorno físico de pruebas y el entorno de producción, así como las herramientas y recursos de que dispone el equipo de prueba. El entorno físico incluye hardware, software y configuraciones de red.
- Identificar los requisitos no Funcionales:
  Determinar el tiempo de respuesta, el rendimiento, la utilización de los recursos y los objetivos y limitaciones.
- Planificar y diseñar las pruebas:
  Identificar los principales escenarios, determinar la variabilidad de los usuarios y la forma de simular esa variabilidad, definir los datos de las pruebas, y establecer las métricas a recoger.
- Configurar el entorno de prueba:
  Preparar el entorno de prueba, herramientas y recursos necesarios para ejecutar cada una de las estrategias, así como las características y componentes disponibles para la prueba.
- Ejecutar la prueba:
  Ejecutar y validar las pruebas, los datos de las pruebas, y recoger los resultados.
- Analizar los resultados:
  Analizar y compartir los resultados de la prueba tanto individualmente, como con un equipo multidisciplinario

A. Herramientas Automatizadas

Las herramientas automatizadas no automatizan todo el procedimiento del testing, pero si ayuda a automatizar partes del mismo. No es un proceso complejo pero sí necesario en cada uno de los niveles de desarrollo de software. Varias son las ventajas por las cuales tanto el equipo de calidad y de desarrollo del software, necesitan adoptar esta política.

- Confiable:
  Las pruebas realizan exactamente las mismas operaciones cada vez que se ejecutan, de tal modo se elimina en gran medida el error humano.
- Repetible:
  Se puede probar cómo el software reacciona bajo la ejecución repetida de las mismas operaciones.
- Programable:
  Se pueden programar las pruebas de manera sofisticada y automatizada.
- Entendible:
  Se puede construir una prueba que cubra cada característica del uso de la aplicación.
  Reutilizable:
  Se puede reutilizar pruebas en diversas versiones, aun realizando cambios en la interfaz de usuario.
- Software de una calidad mejor:
  Porque se pueden funcionar más pruebas en menos tiempo y con pocos recursos.
- Rápido:
  Usuarios humanos son mucho más lentos que usuarios automatizados, por tanto el funcionamiento de las pruebas en herramientas automatizadas se realiza de manera más rápida.
- Reducción de costos:
  Como el número de los recursos para la prueba se reduce. Las inversiones asociadas disminuyen.

Dentro de las herramientas a utilizar se propone el Jmeter. Este es un proyecto de Apache que puede ser utilizado como una herramienta de prueba para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. Utilizar JMeter en aplicaciones web para la comprobación de los recursos del sistema. 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 [5].

B. Gestión de incidencias generadas.

A través de la ejecución de los test son generadas las incidencias o No Conformidades, las cuales se han clasificado en diferentes categorías, ellos son:

- Tiempos de Respuesta.
- Concurrencia de Usuarios.
- Errores HTTP:

Los códigos HTTP describen el estado de un URL cuando un visitante intenta accederlo. Esto incluye mensajes de error que aparecen cuando un visitante encuentra problemas en un sitio web. Los códigos son números de tres dígitos. Hay cinco clases de códigos HTTP de error. El primer dígito determina su categoría [6]:

- Un primer dígito de 1, 2 ó 3 representa una solicitud funcional. Puede encontrar información sobre estos códigos en este artículo.
- Un primer dígito de 4 representa un error del lado del cliente. Los códigos más comunes se extienden de 400a 404.
- Un primer dígito de 5 representa un error del lado del servidor. Los códigos comunes se extienden de 500 a 510.

C. Base de Casos de posible soluciones de errores HTTP.

La elaboración de esta Base de Caso se realizó en teniendo en cuenta la dificultad de varios clientes (*equipos de desarrollo de Software*) de no presentar soluciones concretas antes los errores http generados en estos tipos de testing. Vale destacar que la propuesta que se describe ha sido desarrollada y desplegada en una empresa que brinda servicios de testing, tanto a clientes nacionales como extranjeros. A continuación se detalla una pequeña porción de esta Base de Caso.

Tabla I. DESCRIPCIÓN DE CÓDIGOS

Código	Descripción
400	Solicitud incorrecta. La solicitud contiene sintaxis errónea.
403	Prohibido. La solicitud fue legal, pero el servidor se rehúsa a responderla.
404	No encontrado Recurso no encontrado. Se utiliza cuando el servidor web no encuentra la página o recurso solicitado
405	Método no permitido. Una petición fue hecha a una URI utilizando un método de solicitud no soportado por dicha URI; por ejemplo, cuando se utiliza GET en una forma que requiere que los datos sean presentados vía POST, o utilizando PUT en un recurso de sólo lectura.
500	Error interno. Emitido por aplicaciones embebidas en servidores web, las mismas que generan contenido dinámicamente, por ejemplo aplicaciones montadas en IIS o Tomcat, cuando se encuentran con situaciones de error ajenas a la naturaleza del servidor web.
509	Límite de ancho de banda excedido. Este código de estatus, mientras que es utilizado por muchos servidores, no es oficial.

IV. EVALUACIÓN DE LA EFICIENCIA

Para evaluar la eficiencia en un producto de software se han desarrollado un conjunto de métricas que indican qué porciento de eficiencia presenta una aplicación web convirtiéndose el valor de esta EF (dada en por ciento) en un Indicador a obtener. Dentro de las diferentes métricas propuestas para las evaluaciones se encuentras las relacionadas con las Incidencias o No Conformidades generadas en el proceso de Testing. Se deben tener en cuenta para lograr una evaluación más exacta que se realicen los tres tipos de prueba.

El resultado de la Evaluación Final (EF) se calcula de la siguiente manera:

- Evaluación Final (EF)

  EF = (%TR+%C+%EHTTP)/3 (1)

  Donde

  %TR=*Por Ciento de Tiempo de Respuesta*.

  %C=*Por Ciento de Concurrencia.*

  %EHTTP= *Por Ciento de Errores HTTP.*

Para calcular los anteriores por cientos se utilizarán las siguientes métricas:

- Por Ciento de Tiempo de Respuesta. (%TR)

  %TR= (TRR*100)/TRE (2)

  Donde

  TRR= Tiempo de Respuesta Real.

  TRE= Tiempo de Respuesta Esperado.

- Por Ciento de Concurrencia. (%C)

  %C= (CR*100)/CE (3)

  Donde

  CR= Concurrencia Real.

  CE= Concurrencia Esperada.

- Por Ciento de Errores HTTP. (%HTTP) (4)

  %HTTP= (EHTTP*100)/CURL

  Donde

  EHTTP: Errores HTTP generados.

  CURL: Cantidad de URL visitadas.

Para expresar los indicadores obtenidos se asociarán a los tipos de test o sub-características de la Norma.

Tipos de Test/Sub-características	Indicadores Involucrados
Rendimiento	%TR, %C
Carga	%C, %HTTP
Estrés	%C, %HTTP

Nota: Se está actualizando la propuesta de evaluación para además de tener un resultado cuantitativo, lograr uno cualitativo se realizando con datos históricos de procesos de Testing.

V. CONCLUSIONES

Con la presente investigación se logró realizar la evaluación de la característica de Eficiencia de la norma ISO/IEC 9126, obteniéndose indicadores, conllevando a una mayor organización dentro de las pruebas de rendimiento, estrés y carga.

VI. REFERENCIAS

[1] ISO-"NC-ISO-IEC-9126-1-Ingeniería de software-Calidad del producto-Parte 1: Modelo de calidad". Oficina Nacional de Normalización, ONN. Disponible en: http://www.iso.org/iso/home.html (2005)

[2] B. Hetzel. "The complete guide to Software Testing, 2nd Edition",QED Information Sciences Inc, (1988).

[3] E . Dijkstra. "Notes on structures Programming".TH Report 70 –WSK-03. Disponible en: http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.pdf. (1970).

[4] Testing de performance, carga, stress y volumen, Centro de Ensayos de Software (CES). Disponible en: http://www.ces.com.uy/index.php/servicios/testing-independiente/testing-de-performance. (2014).

[5] Apache JMeter™."The Apache Jakarta Project". Disponible en: http://jakarta.apache.org/jmeter/. (1999-2013).

[6] Status Code Definitions. Disponible en: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html(2014)

Delvis Echeverría Perez, graduado de Ingeniero en Ciencias Informática en 2007, y Máster en Calidad de Software en 2011 por la Universidad de las Ciencias Informáticas. Se ha desempeñado como Especialista en Pruebas de Software desde 2008 en el Centro Nacional de Calidad de Software (CALISOFT). Es profesor en la Universidad de las Ciencias Informáticas desde 2007.

Ariannis Abella Paumier, graduada de Ingeniera en Ciencias Informática en 2013, por la Universidad de las Ciencias Informáticas. Se ha desempeñado como Especialista Auditorias y Revisiones de software desde el 2013 en el Centro Nacional de Calidad de Software (CALISOFT).