Análisis e implementación de un Toolkit para Testing de Performance

Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Montevideo - Uruguay

Marzo 2005

Informe de Proyecto de Grado presentado al
Tribunal Evaluador como requisito de graduación
de la carrera Ingeniería en Computación

Autores: Ignacio Abel
Pablo Giampedraglia

Tutor: Gustavo Vázquez

Resumen

Este proyecto fue motivado por el Centro de Ensayos de Software (CES), en particular por el laboratorio de Ensayos de Plataformas. Entre las principales tareas de este laboratorio, se encuentra la de realizar Testing de Performance, el cuál consiste en simular el uso de una aplicación y determinar las fronteras operacionales. El principal objetivo del proyecto es la elaboración de un Toolkit para Testing de Performance que permita a los usuarios realizar dicha actividad de forma más eficaz, eficiente y simple. El Toolkit planteado por este proyecto está formado por un producto llamado "Performance Center" y aplicaciones existentes en el mercado para Testing de Performance. Entre las motivaciones de implementar el producto "Performance Center" se encuentran el tener un único repositorio de datos que facilite el análisis de los mismos y la posibilidad de tener un control centralizado en escenarios de testing en los que participen varias herramientas.

Para lograr alcanzar el objetivo propuesto se establecieron dos objetivos adicionales, el primero es realizar un estudio del estado del arte sobre el área y el segundo es evaluar herramientas para Testing de Performance. El estudio del estado del arte del área fue planteado debido a la poca bibliografía que hay sobre los puntos teóricos y metodológicos del área. El estudio tuvo como resultado un artículo con la definición de los conceptos relacionados y un resumen de las técnicas que se utilizan. Para la selección de las herramientas que conformarían el Toolkit fue necesario realizar una evaluación de las herramientas para Testing de Performance existentes en el mercado. Dicha evaluación fue realizada siguiendo una metodología y se obtuvo como resultado una Tabla Comparativa que resume los datos más importantes de cada herramienta.

"Performance Center" está desarrollado en Java, tiene una interfaz Web y utiliza una base de datos relacional para almacenar toda la información recolectada. El sistema permite crear tests dentro de una estructura de árbol, ejecutar dichos tests en diferentes herramientas así como la posterior obtención y transformación de los resultados a un formato estándar. Entre sus funcionalidades se encuentran: la ejecución de consultas SNMP, la visualización de los resultados en forma grafica, la administración de usuarios y la exportación e importación de resultados y tests. "Performance Center" presenta un diseño en capas con una arquitectura que permite la inclusión de Plugins. Los nuevos Plugins permiten a "Performance Center" extender sus funcionalidades a nuevas herramientas de Testing de Performance, permitiendo la definición de tests, su ejecución y la obtención de los resultados de la ejecución en las nuevas herramientas. Este mecanismo de extensibilidad se diseñó especialmente para lograr la integración de herramientas de forma simple. Entre las tecnologías utilizadas para realizar la aplicación se encuentran: MySQL (base de datos), Castor (data mapper), Apache Tomcat (servidor Web), Axis (Web Services) y JFreeChart (gráficas).

Palabras Claves en Ingles:
Testing, Performance, Stress, Load, Scalability, Toolkit.

Palabras Claves en Español:
Verificación, Desempeño, Estrés, Carga, Escalabilidad.

Tabla de contenido

CAPÍTULO 1: INTRODUCCIÓN

1.1 DESCRIPCIÓN:

El presente documento es el informe del proyecto *"Análisis e implementación de un Toolkit para Testing de Performance"* de la asignatura Proyecto de Grado del Instituto de Computación. El objetivo principal del proyecto es la elaboración de un producto que permita utilizar de forma centralizada distintas herramientas para Testing de Performance. El proyecto se encuentro dentro del área de Concepción de Sistemas de Información (CSI), que se interesa en la formación, elaboración y aplicación de técnicas que permitan crear, mantener y mejorar los sistemas de información de las organizaciones. [1]

A continuación se presenta brevemente la motivación del proyecto en la cual tuvo un papel muy importante el Centro de Ensayos de Software (CES). En la sección tres se presentan los objetivos del proyecto, en la sección cuatro se presentan los resultados esperados, luego en la sección cinco se describe el CES por ser el cliente del proyecto, en la seis se presentan las conclusiones y los resultados obtenidos y finalmente en la sección siete se detalla la estructura del resto del documento.

1.2 MOTIVACIÓN:

Este proyecto fue motivado por la aparición en el Uruguay del Centro de Ensayos de Software (CES). El CES es una respuesta de la industria del software local a los problemas que enfrenta relacionados con la calidad de sus productos. Uno de estos problemas es la necesidad por evaluar y garantizar la calidad de sus productos que concierne al área de la verificación del software, la cual es un área en desarrollo que presenta grandes desafíos.

La verificación del software está siendo un tema de gran actividad e interés debido a un conjunto de tendencias que están convergiendo. Desde los comienzos del desarrollo de software la evaluación de la calidad del mismo planteó problemas teóricos y prácticos difíciles de resolver. El sostenido aumento en la capacidad de cómputo y almacenamiento de los equipos, hasta el momento no ha hecho más que empeorar la situación. Existe por otro lado una dependencia creciente de los individuos, las organizaciones y de la sociedad respecto al software, lo que genera demandas también crecientes por obtener y asegurar mejores niveles de calidad [2].

Dentro de la verificación del software se encuentra el área de los Test de Performance, que surge de la necesidad de verificar que los sistemas se comportarán de la forma esperada cuando se los someta a cargas reales en ambientes de producción. Esta área tiene cada vez un papel más importante dentro de la verificación.

La primera razón de su importancia es por el aumento de la cantidad de aplicaciones distribuidas. Los desafíos y riesgos entorno a estas aplicaciones son: el gran número de usuarios concurrentes que deben manejar, los tráficos de red a los cuales están sometidas y los grandes volúmenes de datos que manipulan. La segunda razón son las nuevas arquitecturas, por ejemplo cliente-servidor o peer-to-peer, que cada vez incrementan más el número de capas y componentes que deben interactuar para el correcto funcionamiento del sistema.

La realización de los Tests de Performance no es simple porque a los desafíos que presentan el área de verificación de software y las nuevas aplicaciones informáticas, hay que sumarle los que presenta la propia área. Una primera dificultad es la imposibilidad de realizar Testing de Performance sin la ayuda de una herramienta que por lo menos simule cargas de trabajo y monitoree los recursos del sistema. Una segunda dificultad es encontrar herramientas adecuadas

para el sistema a testear ya que en la actualidad hay una gran cantidad de herramientas con las más diversas características. Una tercera dificultad es que en los Tests de Performance, la información producida proviene de distintas fuentes, por ejemplo de las herramientas de testing o utilidades de los sistemas operativos. Debido a esta situación es conveniente tener un control centralizado de dichas fuentes y un único repositorio donde almacenar toda la información recolectada.

Considerando los problemas planteados vinculados al Testing de Performance, el Centro de Ensayos de Software propuso este proyecto que tiene como objetivo analizar, diseñar e implementar un Toolkit para el Testing de Performance.

1.3 OBJETIVOS:

El objetivo del proyecto es la elaboración de un Toolkit para Testing de Performance. Un Toolkit es simplemente un conjunto de aplicaciones (artefactos de software) que los usuarios encargados de realizar Tests de Performance pueden utilizar para llevar adelante su trabajo de forma más eficaz, eficiente y simple. Para lograr cumplir este objetivo el Toolkit tendría que cumplir una serie de requerimientos que se describen en el capítulo tres de este documento. Estos requerimientos determinan que el Toolkit debería presentar las siguientes funcionalidades:

- Crear y mantener datos de pruebas.
- Generar carga al sistema.
- Ejecutar las transacciones.
- Monitorear los recursos del sistema.
- Registrar y analizar los resultados obtenidos.

El objetivo del proyecto es lograr obtener un toolkit que presente todas estas funcionalidades de la forma más integrada posible.

Para poder alcanzar el objetivo del proyecto, es necesario realizar un estudio del estado del arte sobre los Tests de Performance. El proyecto no continúa ninguna línea de trabajo anterior por lo que es preciso realizar una búsqueda muy extensa de bibliografía para elaborar un marco teórico donde se pueda fundamentar todo el trabajo posterior. El estudio del estado del arte tiene que incluir la definición de los conceptos involucrados y de las técnicas utilizadas.

Otra actividad necesaria para alcanzar el objetivo del proyecto es la realización de un estudio sobre herramientas para Testing de Performance y arquitecturas de monitoreo existentes. Dicha actividad es el primer paso para la elaboración de un Toolkit, ya que para obtener un Toolkit de calidad es necesario conocer que funcionalidades ofrecen los distintos productos existentes. A su vez, un aspecto interesante de realizar dicho estudio es el poder obtener conclusiones sobre las populares herramientas open source y de las cada vez más sofisticadas herramientas comerciales. En general no es simple la elección entre herramientas comerciales y open source, porque las comerciales implican una inversión de capital muy alto y las open source un riesgo en respaldo y soporte.

1.4 RESULTADOS ESPERADOS

El principal resultado esperado de este proyecto es la implementación de un Toolkit para realizar Testing de Performance. En la sección de objetivos se enunciaron las funcionalidades que debería presentar el Toolkit y en el capítulo tres se explican todas en detalle. Debido a la duración acotada del proyecto es imposible realizar un Toolkit que presente todas estas funcionalidades perfectamente integradas. Por esta razón se realizó una selección de las características más importantes que debería tener el producto a desarrollar para lograr una integración adecuada de las funcionalidades. El resultado de esta selección fue la siguiente lista de características:

- Permitir el monitoreo de los recursos del sistema.
- Centralizar toda la información producida durante los Tests de Performance en un único repositorio de datos bajo un mismo formato.
- Facilitar el análisis y comparación de los datos recolectados.
- Permitir la extracción de los datos del repositorio en formatos Standard para ser analizados en otras herramientas.

La explicación de porque se decidió implementar un producto con estas características se puede encontrar en el capítulo tres. Por una lista más detallada de requerimientos se puede consultar el Documento de Requerimientos anexado a este informe.

Entre los resultados secundarios, derivados de las actividades realizadas para cumplir con el resultado presentado anteriormente, se encuentran realizar un informe del estado del arte del Testing Performance, que incluya la definición de los conceptos involucrados y un resumen de las técnicas utilizadas. Este informe corresponde al capítulo 2 de este documento.

Otro resultado secundario es la elaboración de un informe sobre el estudio y comparación de herramientas comerciales y open source existentes. Este informe debe contener una tabla que resuma los resultados de la evaluación y las conclusiones obtenidas. El informe corresponde al capítulo 3 de este documento.

1.5 CENTRO DE ENSAYOS DE SOFTWARE:

El CES es una iniciativa que comenzó en el año 2004, y tiene como objetivo brindar servicios de verificación y evaluación de software, funcional y no funcional. El CES es auspiciado por la Unión Europea, el Programa de Naciones Unidas para el Desarrollo, la Universidad de la República (UDELAR) de Uruguay y la Cámara Uruguaya de Tecnologías de la Información [2] [4].

El CES está compuesto por un Laboratorio de Testing, enfocado en la evaluación de productos desde el punto de vista funcional y un Laboratorio de Ensayos de Plataformas que realizará pruebas de desempeño y asistirá a la industria en resolver problemas de funcionamiento en arquitecturas de hardware y software complejas. A estos dos laboratorios se agrega un Observatorio Tecnológico, con el objetivo de escudriñar las novedades y tendencias en el área de Tecnologías de la Información para facilitar el acceso a las mismas por parte de la industria [2] [4].

El proyecto *"Análisis e implementación de un Toolkit para Testing de Performance"* se encuentra enmarcado dentro del Laboratorio de Ensayos de Plataformas. Dentro de las actividades de dicho laboratorio, se encuentran definidos los Tests de Carga, Tests de Stress, Tests de Escalabilidad y Tests de Configuración [4].

1.6 CONCLUSIONES Y RESULTADOS OBTENIDOS

En está sección se resumen las conclusiones y los resultados obtenidos del proyecto para que el lector logre un perspectiva global. Todos ellos se irán desarrollando a lo largo del documento.

El primer resultado fue la implementación de una aplicación llamada "Performance Center" que es parte del Toolkit propuesto. Este producto está detallado en el capítulo cuatro. El segundo resultado fue la elaboración de un artículo sobre Testing de Performance que se encuentra anexado a este documento. En el capítulo dos se presenta un resumen de este artículo. El tercer resultado es un estudio de herramientas de Testing de Performance que se presenta en el capítulo tres. Anexado a este documento se encuentra una "Tabla Comparativa" que resume los resultados de dicho estudio.

En resumen las conclusiones del proyecto son:
- Existe una escasa cantidad de trabajos realizados entorno a los conceptos teóricos y metodológicos del Testing de Performance.
- Es muy importante seleccionar cuidadosamente el conjunto de herramientas a utilizar para realizar un Test de Performance.
- Disponer de una herramienta que centralice los datos recolectados durante la ejecución de los tests es una gran ayuda para su posterior análisis.

1.7 ORGANIZACIÓN DEL DOCUMENTO

Este documento está estructurado en 6 capítulos, en los cuales se pretende presentar de forma resumida el trabajo realizado durante el transcurso del proyecto.

Los capítulos presentes en el documento son suficientemente independientes como para que el lector los pueda abordar en el orden de su preferencia. Se recomienda al lector no familiarizado con la verificación del software leer primero el capítulo 2 para lograr entender el resto de los capítulos.

El presente documento está estructurado en los siguientes capítulos:
- **Capítulo 2:** Se presenta el estudio realizado sobre el estado del arte del Testing de Performance; se explican los conceptos relacionados y las técnicas utilizadas en el área.
- **Capítulo 3:** Se presentan los requerimientos del Toolkit para Testing de Performance, luego se presenta la metodología utilizada para el análisis de las herramientas de Testing de Performance y los resultados obtenidos de la metodología.
- **Capítulo 4:** Se describen las funcionalidades de la aplicación "Performance Center", así como las decisiones tomadas referentes a su arquitectura y sus componentes.
- **Capítulo 5:** Se presenta el nivel de calidad logrado, comentarios sobre las pruebas realizadas y los resultados obtenidos.
- **Capítulo 6:** Se presentan las conclusiones obtenidas del trabajo realizado y se sugieren líneas de trabajo futuro.
- **Referencias:** Se detallan todas las referencias bibliográficas que se citaron en este informe.

Además el presente documento está acompañado de una serie de anexos que permiten profundizar temas tratados en el informe. Los anexos son:
- **Estudio sobre Testing de Performance:** Artículo sobre los conceptos y técnicas del Testing de Performance.
- **Tabla Comparativa:** Muestra de forma resumida y esquemática la información de las distintas herramientas.
- **Diseño del Sistema:** Contiene los documentos Descripción de la Arquitectura y Modelo de Datos.
- **Manual de usuario:** Describe como utilizar el sistema "Performance Center".
- **Manual Técnico**: Esta compuesto por el Documento de Estándares de implementación, la Guía para la extensión del Sistema que describe como crear un Plugin, un Caso de Estudio y por el Javadoc de la aplicación (sólo en versión digital).
- **Documentación del Nivel de Calidad:** Contiene todos los documentos relacionados con el nivel de calidad, estos son: Documento de Requerimientos, Modelo de Casos de Uso, Documento de Trazabilidad, Casos de Prueba de Requerimientos, Plan de pruebas del sistema, Casos de Prueba con Datos, Plan de Verificación y Validación, Reporte de Fallas, Pruebas de Aceptación, Plan del Proyecto y clases de prueba JUnit.

CAPÍTULO 2: TEST DE PERFORMANCE

2.1 INTRODUCCIÓN

En este capitulo se definirá Test de Performance y los conceptos importantes relacionados.

El Test de Performance ha evolucionado en los últimos años y se ha convertido en un componente crítico del proceso de prueba, cada vez es más importante para que los grandes proyectos de software no fracasen. El desempeño en muchos casos es fundamental para el éxito de una aplicación. Por ejemplo si una aplicación de e-business brinda un servicio con tiempos de respuesta demasiado largos, los clientes la dejaran de usar y encontraran un servicio alternativo.

2.2 TEST DE PERFORMANCE

El Test de Performance simula el uso verdadero de la aplicación y determina las fronteras operacionales.

El objetivo primario del Test de Performance según Gerrard y O'Brien es: *"Demostrar que el sistema funciona contra la especificación con tiempos de respuesta aceptables mientras procesa los volúmenes de transacciones requeridos con una base de datos de producción."* [5]

Los principales componentes de los Tests de Performance son una infraestructura para poder correr pruebas automatizadas por largos periodos de tiempo y los scripts, que son las entradas de esos programas. Los Tests de Performance no pueden ser hechos manualmente ya que es sumamente difícil que una persona pueda generar una carga significativa y sostenida. Es inevitable usar alguna clase de generador de carga automático.

Un Test de Performance no sólo determina el desempeño de una aplicación. Según Gerrard y O'Brien también se realizan con los siguientes objetivos [5]:

- *Identificar puntos débiles de la arquitectura:* La carga del test puede ser incrementada hasta niveles extremos para provocar fallas en el sistema y así detectar cuales son los cuellos de botella y componentes débiles de la arquitectura.
- *Detectar bugs ocultos en la aplicación:* Los tests que corren por largos periodos pueden causar fallas, por ejemplo debido a pérdidas de memoria. De está forma se pueden descubrir problemas o conflictos difíciles de detectar con testeo funcional.
- *Poner a punto el sistema (Tuning):* Las corridas repetidas de los tests pueden ser utilizadas para optimizar el sistema. En cada corrida se pueden hacer cambios al sistema y verificar que esos cambios (Tuning) estén teniendo el efecto deseado, que es mejorar el desempeño.
- *Verificar la confiabilidad del sistema:* La ejecución de Tests de Performance por periodos prolongados, es la única manera de evaluar la confiabilidad del sistema.
- *Asegurar la capacidad de crecimiento del sistema:* Correr tests variando los recursos del sistema es la única forma de evaluar la escalabilidad del mismo.

El Test de Performance puede ser dividido en:

- Test de Carga
- Test de Estrés
- Test de Escalabilidad

A continuación explicaremos cada tipo de test en detalle.

2.3 TEST DE CARGA

Es el proceso de analizar las aplicaciones de software y su infraestructura de soporte (como una base de datos) para determinar el desempeño y la capacidad de manejar transacciones. El análisis se realiza por medio del modelado y la simulación de condiciones que representen el uso verdadero de la aplicación.

El Test de Carga determina el comportamiento de un sistema bajo varias cargas de trabajo. El objetivo es determinar como los componentes del sistema reaccionan al aumentar gradualmente las cargas de trabajo.

En el siguiente fragmento Boris Beizer nos describe de forma muy clara que significa Test de Carga:

"El Test de Carga somete a un sistema a una carga estadísticamente representativa. [...] En Test de Performance, la carga se varía de un mínimo (cero) hasta el nivel máximo que el sistema puede sostener sin quedar sin recursos o tener transacciones sufriendo (dependiendo de la aplicación) demoras excesivas. Otro uso del término de Test de Carga es como una prueba cuyo objetivo es determinar la carga máxima sostenible que el sistema puede manejar. En este uso, Test de Carga es meramente testear con las tasas más altas de llegada de transacciones." [6]

Es importante destacar que en un Test de Carga el sistema debe responder al nivel de carga sometido de forma adecuada, en otras palabras según sus requerimientos. Si el sistema no responde de forma esperada, los resultados obtenidos no corresponden a un Test de Carga, sino a uno de Estrés.

2.4 TEST DE ESTRÉS

El Test de Estrés se realiza para evaluar un sistema o componente en los límites especificados en sus requerimientos, o más allá de ellos. Determina el punto en que un sistema comienza a brindar un desempeño inaceptable. Este punto puede ser llamado punto de quiebre.

Boris Beizer define este concepto de la siguiente manera:

"Test de estrés es someter a un sistema a una carga no razonable mientras se le niegan los recursos (por ejemplo, memoria RAM, disco, mips, interrupciones, etc.) necesarios para procesar esa carga. "[6]

La idea es estresar a un sistema al punto de quiebre para encontrar errores que podrían ser potencialmente perjudiciales. Esta es la única forma de poder detectar este tipo de errores. No se espera que el sistema procese la sobrecarga sin los recursos adecuados, pero si que se comporte de una manera razonable. Por ejemplo, una aplicación podría entrar en un modo de fallo, y evitar corromper o perder datos.

Los errores o modos de fallo descubiertos bajo pruebas de estrés pueden o no ser reparados dependiendo de la aplicación, el modo de fallo, las consecuencias, etc. En conclusión, la carga en pruebas de estrés es conducida deliberadamente a forzar el agotamiento de los recursos del sistema.

2.5 TEST DE ESCALABILIDAD

El Test de Escalabilidad permite determinar si un sistema puede efectivamente trabajar con cargas de trabajo variables cuando se le brindan los recursos necesarios. En otras palabras el Test de Escalabilidad evalúa los efectos de agregar hardware y/o software adicional para distribuir el trabajo entre componentes del sistema.

Las pruebas de escalabilidad se realizan en una variedad de configuraciones, con variables tales como la velocidad de la red, número y tipo de servidores o CPU's, memoria, etc.

Según Elfriede Dustin los objetivos del Test de Escalabilidad son [7]:

- *"Verificar que la arquitectura entregue un alto grado de escalabilidad, permitiendo al número de usuarios y a los volúmenes de datos del sistema aumentar sin sacrificar funcionalidad o desempeño. La escalabilidad en nuestro contexto es entendida como la habilidad del sistema de manejar una demanda creciente de usuarios."*

- *"Verificar que la arquitectura del sistema permita la distribución de un número creciente de usuarios concurrentes cuando se cambia la cantidad y/o capacidad del hardware empleado."*

2.6 EL PROCESO DE TESTEO DE PERFORMANCE

Existen cuatro actividades principales en el Testeo de Performance. Una quinta actividad "puesta a punto", acompaña las actividades del tester y es normalmente realizado por especialistas técnicos. La puesta a punto puede ser comparada con la actividad de corregir defectos luego de realizar las pruebas funcionales. La puesta a punto puede involucrar cambios en la infraestructura de la arquitectura, pero usualmente no afecta la funcionalidad del sistema bajo test. Un esquema del proceso se puede ver en la siguiente figura.

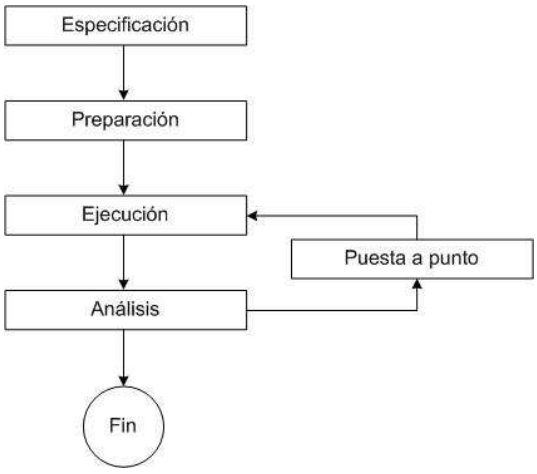

Figura 1: Esquema del proceso del Testeo de Performance [8]

La siguiente tabla muestra las actividades que usualmente se realizan en cada etapa del proceso:

Especificación	• Documentación de los requerimientos de performance: * Volúmenes de bases de datos * Perfiles de carga * Requerimientos de tiempos de respuesta • Preparación de un calendario con los perfiles de carga a ser testeados. • Inventario de las transacciones del sistema que comprendan las cargas a ser testeadas. • Inventario de las transacciones del sistema a ser ejecutadas y tiempos de respuesta medidos. • Descripción del análisis y de reportes a generar.
Preparación	• Preparación de una base de datos de prueba con volúmenes apropiados. • Generación de scripts de transacciones del sistema para probar esa carga. • Generación de scripts de transacciones del sistema cuyos tiempos de respuesta serán medidos (posiblemente los mismos que los anteriores). • Definir las de cantidades de carga (implementación de perfiles de carga). • Preparación de datos de prueba para parametrizar los scripts automáticos.
Ejecución	• Ejecución de tests provisorios. • Ejecución de tests de performance. • Repetir las corridas de tests cuando sea necesario.
Análisis	• Recolectar y guardar resultados de los tests. • Preparación de análisis gráfico y tabular. • Preparación de reportes, incluyendo interpretación y recomendaciones.
Puesta a punto	• Mejoras en la performance del software, middleware y organización de la base de datos de la aplicación. • Cambios a los parámetros del servidor. • Mejoras en hardware del cliente, servidor, red o enrutamiento.

Tabla 1: Actividades del proceso de Test de Performance [8]

Para mas información acerca de este proceso recomienda la lectura de [8].

CAPÍTULO 3: REQUERIMIENTOS DEL TOOLKIT Y ESTUDIO DE HERRAMIENTAS

3.1 INTRODUCCIÓN

La primera parte de este capítulo tiene como objetivo describir los requerimientos del Toolkit para Testing de Performance. De éstos requerimientos se desprenden las decisiones tomadas con respecto a las aplicaciones que integrarían el Toolkit. Una de estas decisiones fue conformar el Toolkit con herramientas de Testing de Performance existentes en el mercado. Por lo tanto se decidió realizar un estudio de este tipo de herramientas para ayudar a seleccionar las que serían parte del Toolkit.

La segunda parte de este capítulo tiene como objetivo presentar la metodología seguida para el estudio de las diferentes herramientas para Testing de Performance existentes y presentar los resultados de dicho estudio. El estudio abarcó herramientas comerciales y no comerciales, para testear servidores Web y para testear servicios en general. El resultado principal de este estudio fue una "Tabla comparativa" que reúne las principales características de cada herramienta y breves comentarios sobre los que se consideraron los puntos a destacar de cada una. El resultado secundario del estudio fue el conocimiento adquirido durante su realización lo cual permitió realizar el posterior diseño del Toolkit.

La estructura del capítulo es la siguiente: en la segunda sección se describen los requerimientos del Toolkit, en la tercera sección se presenta el proceso que se siguió para testear las herramientas y en la cuarta sección se describen los resultados que se obtuvieron en la aplicación de cada paso.

3.2 REQUERIMIENTOS DEL TOOLKIT PARA TESTING DE PERFORMANCE

Para realizar un Test de Performance es necesario un conjunto de aplicaciones al cual denominamos Toolkit. El conjunto de las aplicaciones que agrupa un Toolkit deben cubrir ciertas características y funcionalidades para que el Toolkit permita realizar Tests de Performance. Un Toolkit eficaz debería tener: [8]

a. Utilidades para crear y mantener los datos de pruebas:
En los Tests de Performance es necesario crear un gran volumen de datos. Estos datos se almacenan en general en bases de datos relacionales para acceder por medio de SQL.

b. Herramientas para generar carga al sistema:
Estas herramientas manejan los clientes virtuales y generan las transacciones de los tests.

c. Herramientas para ejecutar las transacciones:
Estas herramientas manejan la interfaz de la aplicación y miden los tiempos de respuesta. En la mayoría de los casos son las mismas herramientas que para generar la carga, pero algunas tienen procesos diferentes para la generación de la carga y la ejecución de los tests.

d. Monitores de los recursos del sistema:
Estas herramientas monitorean los recursos del sistema de los clientes y de los servidores. Los recursos pueden ser: el porcentaje de uso de procesador, el porcentaje de uso de memoria, el tráfico de la red, la actividad de la base de datos, etc.

e. Utilidades para registrar y analizar los resultados obtenidos:
Las herramientas que ejecutan los tests y los monitores de los recursos del sistema generan una gran cantidad de resultados. Aunque muchas de las herramientas tienen facilidades para

el análisis de los datos es útil combinar los resultados de las distintas fuentes en un único repositorio con los datos resumidos e integrados.

En el diseño del "Toolkit para Testing de Performance" aquí presentado se observó que existían una gran cantidad de herramientas comerciales y open source que cubren muy bien las necesidades de los usuarios referentes a los puntos "a", "b", "c" y "d". Pero se observó que no existía una aplicación que permitiera cubrir las necesidades relacionadas con el punto "e". Por lo general este punto se cubre de forma ad-hoc, es decir mediante hojas de cálculo, procesadores de texto y bases de datos.

Viendo esta realidad se decidió implementar un producto que cubriera las necesidades del punto "e", es decir que integrara los resultados obtenidos durante la ejecución de los tests. El producto además debería cubrir otros requerimientos, los más importantes son integrar herramientas para Testing de Performance y facilitar el monitoreo de los recursos del sistema. Asimismo debido al ambiente en que se utilizaría el producto (centro de ensayos en donde las empresas tercerizan sus necesidades de test) era necesario que permitiera la administración de usuarios y la consulta de resultados de forma remota. El producto implementado se llamó "Performance Center" y está detallado en el capitulo 4.

En resumen se puede decir que el Toolkit para Testing de Performance propuesto está formado por la aplicación "Performance Center" y por un subconjunto de herramientas de Testing de Performance existentes en el mercado. Para poder seleccionar las herramientas que conformarían el Toolkit se elaboró el estudio que se presenta a continuación.

3.3 PROCESO DE EVALUACIÓN DE HERRAMIENTAS DE TESTEO

Para realizar la evaluación de las herramientas de testeo se siguieron los pasos que se detallan a continuación:

a. Identificación de los puntos relevantes a evaluar:
El primer paso fue analizar los puntos y características de las herramientas de testing que serían tenidos en cuenta al momento de evaluar y comparar las mismas. Estos puntos y características corresponden a las columnas de la "Tabla Comparativa". Por ejemplo la plataforma para la cual fue diseñada una herramienta es una característica imprescindible para su evaluación.

b. Identificación de herramientas:
El segundo paso fue identificar herramientas que permitan realizar test de performace y que tengan un nivel mínimo de calidad como para ser utilizadas. Se abarcó un gran número de herramientas para poder realizar una evaluación completa y profunda.

c. Evaluación general de las herramientas:
La evaluación general de las herramientas se realizó estudiando su documentación, y en los casos en que fue posible ejecutando pequeños tests. En este paso se determinaron las características de las herramientas en los puntos relevantes antes seleccionados, es decir, que se completaron las columnas de la "Tabla Comparativa".

d. Selección de las herramientas a evaluar:
El siguiente paso fue seleccionar el conjunto de herramientas que serían evaluadas con profundidad. Las herramientas de este conjunto fueron las que se consideraron más completas, tuvieron un mayor nivel de calidad o tuvieron características interesantes que se desearon evaluar.

e. Prueba de las herramientas:
En esta etapa se procedió a instalar cada herramienta seleccionada en el punto anterior para aprender a utilizarla y realizar una evaluación más profunda. Se realizaron tests de Performance con cada herramienta para analizar su correcto funcionamiento. Fue muy importante identificar errores de implementación que inutilicen funcionalidades de las mismas.

f. Evaluación de los resultados de las herramientas:
El siguiente paso fue evaluar los resultados, reportes y logs que las herramientas proveen. Entre las propiedades que se tuvieron en cuenta están: los tipos de resultados que ofrece la herramienta, la cantidad de datos que se pueden obtener, el nivel de detalle de los resultados, la presentación de los datos y la forma en que los almacena. La forma de almacenar los resultados fue un punto muy importante dentro de la evaluación, debido al contexto en que se realizaba este estudio. Para la elaboración del Toolkit que una herramienta almacene los datos recolectados en un formato estándar es una gran ventaja para su posterior integración. Las herramientas que guardan los resultados obtenidos en formato binario dificultan el acceso a sus datos.

g. Evaluación global:
Se procedió a evaluar el comportamiento de las herramientas probadas en forma global, teniendo en cuenta todos los aspectos. Como resultados se obtuvieron los comentarios que se presentan en la "Tabla Comparativa".

Es importante mencionar que el proceso presentado no fue seguido en forma lineal (cascada), se realizó de forma iterativa, incrementando las herramientas y logrando mayor profundidad en cada iteración.

3.4 RESULTADOS DEL PROCESO DE EVALUACIÓN

a. Identificación de los puntos relevantes a evaluar:
A continuación detallamos los puntos y características que se consideraron relevantes para la revisión preliminar de las herramientas.

Licencia	Tipo de licenciamiento para el uso de la herramienta. Se dividen principalmente en comerciales y libres (Open source).
Plataforma	Sistemas operativos y arquitectura sobre la que corre la herramienta.
Tamaño	Cantidad de MB que ocupa el paquete de instalación de la aplicación.
Interfaz	Tipo de interfaz que ofrece. Puede ser interfaz gráfica – GUI – o interfaz de consola.
Lenguaje	Lenguaje de programación en la que está desarrollada la herramienta.
Ultimo release / Versión	Versión actual de la herramienta y su fecha de liberación.
Tipo de test / Orientación	Clase de pruebas que puede realizar y sobre que tipo aplicación las efectúa.
Requerimientos de Hardware / Software	Establece las exigencias mínimas de hardware que la herramienta necesita para correr. Además los requerimientos del sistema operativo u otro software necesario para el correcto funcionamiento de la herramienta.
Arquitectura	La organización de los distintos componentes de la

	herramienta.
Limitaciones	Restricciones conocidas o detectadas de la herramienta que limiten su uso de alguna manera.
Facilidad de uso	Características detectadas que faciliten o ayuden a la utilización de la misma.
Output	Salida que provee la herramienta. Tipo de presentación de los datos.
Formato de la salida	Formato de los resultados que han sido grabados por la herramienta.

Tabla 2: Puntos relevantes a evaluar de cada herramienta de testing

Todos los puntos anteriores junto a "Nombre de la herramienta" y "Página Web", que identifican a las herramientas, conforman las columnas de la "Tabla Comparativa". También se agregó una columna llamada "Comentarios" donde se registraron los comentarios finales de la evaluación.

b. Identificación de herramientas:

Para encontrar herramientas de testeo de Performance se realizaron búsquedas en la Web y en documentos sobre testing de Performance. Las herramientas de testeo de Performance más populares se encuentran referenciadas en varios documentos sobre el tema y paginas Web. Por ejemplo en el libro "*Risk-Based E-Business Testing"* se encuentra una extensa lista de herramientas [8].

Además, se encontraron documentos en los cuales se realizaban evaluaciones de algunas herramientas, los cuales fueron de gran ayuda para priorizar el orden de evaluación. Por ejemplo "An overview of load test tools" de Julien Buret y Nicolas Droze [9], y "Load Test Tools Evaluation" de Abraham Jacob, Riyaj Shaik y Paul Tennis [10].

También se encontraron sitios relacionados con el testeo de Performance, que poseían vínculos a las paginas de las herramientas más populares, por lo que estas paginas fueron utilizadas como punto de partida para la evaluación de los diferentes productos. Las direcciones Web de dos de estos sitios son:

- http://www.opensourcetesting.org/performance.php [11]
- http://testingfaqs.org/t-load.html [12]

Las herramientas open source seleccionadas fueron:

- The Grinder
- OpenSTA
- Apache Jmeter
- Database Opensource Test Suite
- DBMonster
- Deluge
- Dieseltest
- Hammerhead 2 - Web Testing Tool
- Http_load
- OpenLoad
- Siege
- Stress_driver
- TestMaker
- Web Application Load Simulator
- Web Polygraph

Las herramientas comerciales seleccionadas fueron:

- Microsoft Application Center Test
- Rational Performance Tester
- ANTS - Advanced .NET Testing System
- AutoController
- Benchmark FactoryTM
- Capacity Calibration
- Chariot
- Cyrano
- Inc. Test, FORECAST
- Load Runner (Load Test)

c. Evaluación general de las herramientas:

Para la evaluación de herramientas comerciales en la mayoría de los casos no se realizo la instalación de las mismas, ya que no se disponía de sus licencias ni de sus códigos fuentes, por lo que la información obtenida proviene de la información que publica el fabricante y de algunos artículos independientes. Las herramientas no comerciales, se instalaron y ejecutaron en la mayoría de los casos, además de leerse la documentación que las mismas proveían.

d. Selección de las herramientas a evaluar:

Después de la evaluación general de las herramientas se seleccionó la lista que se presenta a continuación. La primera razón para seleccionar estas herramientas fue su popularidad, ya que están referenciadas en una gran cantidad de sitios y artículos.

- **Apache JMeter:** Fue seleccionada por ser multiplataforma, permitir testear varios tipos de sistemas y pertenecer al proyecto "Apache Jakarta", el cual es muy prestigioso dentro de la comunidad Java.
- **DieselTest:** Es una de las herramientas más pequeñas dentro de las especializadas en sitios Web y fue seleccionada porque se deseó evaluar que prestaciones podía ofrecer una herramienta tan pequeña.
- **Microsoft Application Center Test:** fue seleccionada por ser una herramienta simple y muy difundida, ya que se incluye con algunas distribuciones de la plataforma .NET.
- **OpenSTA:** Fue seleccionada por ser considerada la más completa dentro de las open source para sitios Web.
- **The Grinder 3:** Fue seleccionada por ser una herramienta multiplataforma muy versátil que permite testear todo tipo de sistemas.

e. Prueba de las herramientas:

Luego de instalar todas las herramientas se procedió a realizar distintos tests de Performance con ellas.

- **Apache JMeter:** Se realizaron tests a servidores Web y a base de datos.
- **DieselTest:** Se grabaron scripts y se ejecutaron.
- **Microsoft Application Center Test:** Se grabaron distintos scripts y se ejecutaron variando sus opciones de configuración.
- **OpenSTA:** Se grabaron scripts se realizaron consultas SNMP, consultas NT Performance, se ejecutaron tests con distinta cantidad de usuarios virtuales y se analizaron todos los resultados y graficas que brinda.
- **The Grinder 3:** Se realizaron los tests que se incluyen como ejemplo y se varió la cantidad de procesos, usuarios virtuales e iteraciones.

f. Evaluación de los resultados de las herramientas:

- **Apache JMeter:** Permite definir varios tipos de recolectores de resultados pero su uso es complejo. Los resultados los almacena en formato xml.
- **Diesel Test:** Tiene un conjunto reducido de resultados, no brinda ayuda para analizarlos y los almacena en formato texto.
- **Microsoft Application Center Test:** Presenta los resultados de forma muy resumida y guarda todos los datos en formato xml, lo que es muy bueno porque simplifica su extracción.
- **OpenSTA:** Brinda una gran cantidad de resultados y graficas, lo que facilita el análisis y la comparación de resultados. Todos los resultados de un test se encuentran bajo un mismo directorio, la mayoría en formato texto y algunos pocos en formato binario.
- **The Grinder 3:** Brinda una consola que recolecta los datos de los distintos procesos y clientes virtuales, la interfaz es pobre y los datos recolectados son limitados, aunque se puede programar mediante Jython la recolección de más datos. Los resultados básicos se pueden salvar en formato "csv" y mediante Jython [23] se pueden obtener otros resultados en el formato que se desee.

g. Evaluación global:

En la "Tabla Comparativa" se presentan los comentarios sobre cada una de las herramientas. Aquí mencionaremos los comentarios de las herramientas seleccionadas.

- **Apache JMeter:** Es una herramienta multiplataforma, altamente configurable, compleja, con una interfaz gráfica no muy amigable y que permite realizar tests a diferentes tipos de sistemas.
- **DieselTest:** Es una herramienta muy pobre, demasiado básica y tiene una interfaz gráfica poco atractiva. Además sólo permite realizar tests a sitios Web.
- **Microsoft Application Center Test:** Es una herramienta muy fácil de utilizar, presenta la información de forma muy simplificada y sólo permite realizar tests a sitios Web.
- **OpenSTA:** Es la herramienta más completa entre las open source, tiene una excelente interfaz de usuario, permite definir consultas SNMP y NT Performance, tiene muchos parámetros configurables para la ejecución de los tests y sólo permite realizar tests a sitios Web.
- **The Grinder 3:** Es una buena herramienta aunque está en versión beta. Es muy flexible pero no aporta mucha ayuda para definir los tests ya que está pensada para ser usada por los propios desarrolladores.

CAPÍTULO 4: PERFORMANCE CENTER

4.1 INTRODUCCIÓN

Luego de haber realizado un estudio de lo que un Toolkit para testing de Performance debía ofrecer, se prosiguió con la etapa de desarrollo del mismo. El producto se llama Performance Center (PC) y posee todas las características de un Toolkit que fueron mencionadas en los capítulos anteriores. En este capítulo se describe las funcionalidades generales que PC posee, se explica como sería su uso típico dentro de una organización de testing de Performance, se explica la arquitectura del producto desarrollado y finalmente se expone el conjunto de tecnologías que fueron utilizadas en su desarrollo.

4.2 FUNCIONALIDADES

PC es una herramienta con interfaz Web que centraliza el uso de herramientas para Testing de Performance. Permite la ejecución de tests en las diferentes herramientas que integra así como la posterior obtención y transformación de los resultados a un formato estándar. Entre sus funcionalidades se encuentran: la ejecución de consultas SNMP, la administración de usuarios, la gestión de alarmas y la exportación e importación de resultados y tests. Además, en su interfaz Web, se presenta una estructura de árbol que facilita la navegabilidad y posibilita el manejo de los tests y resultados de forma muy amigable.

Tests: PC permite la generación de tests para diferentes herramientas de Performance y la posterior ejecución en las mismas. Dependiendo de la herramienta, la ejecución puede ser automática o semi-automática. Esto depende de las posibilidades de cada herramienta, por ejemplo API'S o plugins, que determinan que tan automáticamente se ejecutará el test en cada herramienta una vez lanzado desde PC. Un test se asocia a una de las herramientas de Testing de Performace que el sistema integra y contiene un script, consultas SNMP y resultados de los tests y de las consultas SNMP. El concepto de script corresponde a lo que la herramienta de testing va a ejecutar y esta escrito en el lenguaje de la herramienta a la cual esta asociado ese test.

En la figura se ilustra este concepto:

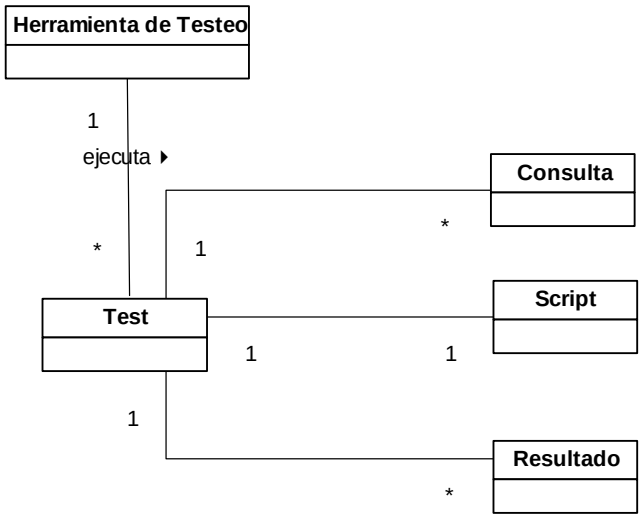

Figura 2: Diagrama del modelado de los Tests

Resultados: Los resultados de los diferentes tests se guardan en la base de datos de PC para su posterior visualización, comparación, análisis y exportación. Los mismos presentan un conjunto de

variables estándar que son únicas para todos los resultados de todas las herramientas que Performance Center integra.

Consultas SNMP: A la vez que PC permite la ejecución de Tests, posibilita la ejecución de consultas SNMP a diferentes hosts. SNMP (Simple Network Management Protocol) es el protocolo estándar de Internet para software de administración de redes. SNMP se utiliza para monitorear dispositivos de red y recolectar datos de Performance de distintas MIBs (Management Information Bases). De esta forma, PC ayuda en el monitoreo del sistema a la herramienta de medición de Performance que se esté utilizando. PC puede obtener toda la información disponible de la MIB del host objetivo mediante consultas SNMP.

Estructura de árbol: Para facilitar la navegación entre los distintos tests se decidió organizar los mismos en una estructura de árbol amigable e intuitiva. Una estructura de árbol tiene la ventaja de permitir agrupar los tests según distintos conceptos lógicos. La estructura de árbol que PC presenta para desplegar su repositorio de tests, consultas SNMP y resultados está organizada mediante directorios y ensayos.

Directorios y Ensayos: Los directorios son un concepto similar al de los directorios de los sistemas operativos más difundidos. Los mismos pueden tener más directorios o ensayos. Los ensayos son un nuevo concepto y tiene la función de agrupar tests de similares características.

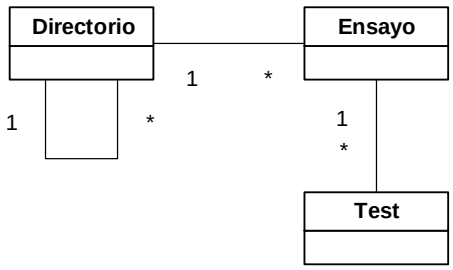

Figura 3: Diagrama de modelado de los Directorios y Ensayos

Importación y Exportación: Es posible importar y exportar resultados o tests. El formato de exportación es XML o CSV y el de importación es XML.

Alarmas: Para la visualización de resultados de herramientas y de resultados de consultas SNMP se proveen alarmas visuales que indican los valores que se encuentran por fuera de un rango definido por el usuario. Las alarmas se establecen para cada variable que presentan los resultados y están compuestas por valores máximos y mínimos admitidos. Si al visualizar un resultado una variable del mismo esta fuera de ese rango, la misma presenta un icono que indica si esta por encima o por debajo del rango preestablecido.

Usuarios: Se definen tres tipos de usuarios: Administradores, Diseñadores y Testers. Todo usuario en el sistema pertenece a alguno de estos tres tipos. PC otorga distintos permisos para acceder a las funcionalidades dependiendo del tipo de usuario. Los usuarios tienen un orden jerárquico, es decir, los usuarios Diseñadores tienen todos los permisos de los usuarios Testers y los usuarios Administradores tienen todos los permisos de los usuarios Diseñadores. Sólo los usuarios administradores tienen privilegios suficientes como para poder crear nuevos usuarios, y los usuarios Testers no tienen privilegios para crear nuevos tests.

4.3 ESCENARIO TÍPICO DE USO:

Típicamente se utilizará el sistema como muestra el diagrama:

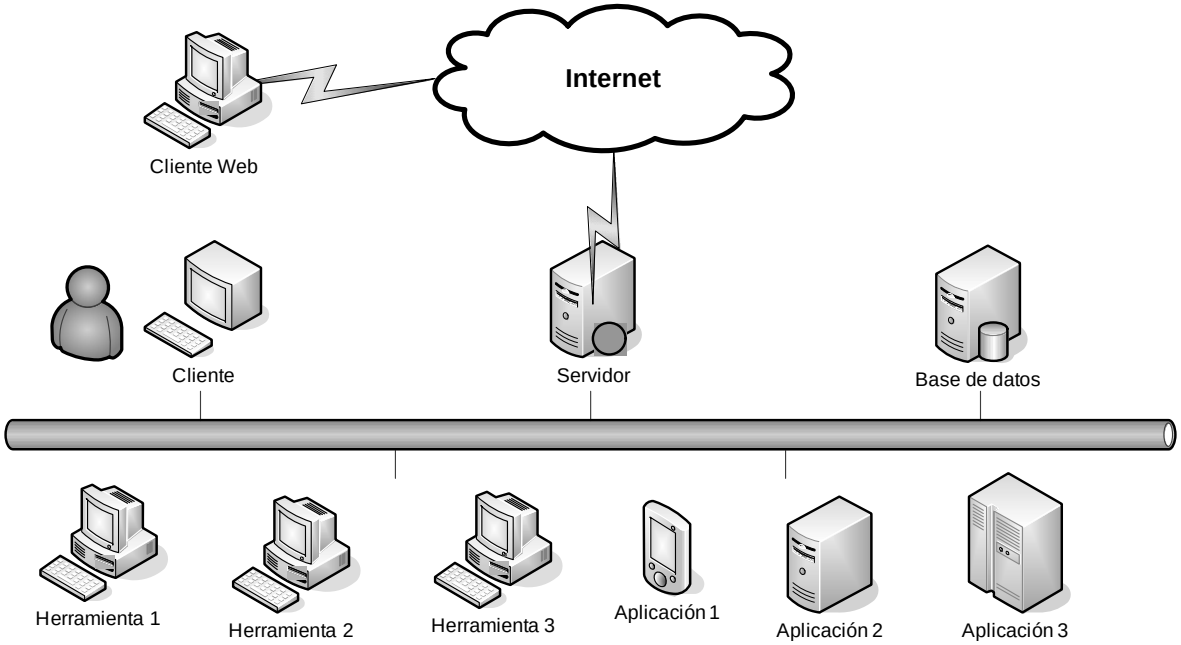

Figura 4: Utilización típica e PC dentro de una institución

En la figura se puede apreciar la arquitectura de la red sobre la cual podría utilizarse PC. En este ejemplo, el servidor es el que contiene la aplicación PC y en la misma red se encuentra la base de datos que PC utiliza. Se pueden apreciar dos tipos de clientes: un cliente de la misma LAN y un cliente Web. Los clientes de la LAN utilizan en general la interfaz HTML que PC brinda y se dedican a administrar y ejecutar los Tests de los sistemas deseados. El ambiente para el cual fue diseñado PC es por ejemplo el brindado por el Centro de Ensayos de Software (CES). En un ambiente de este tipo los clientes Web pueden ser los que enviaron las aplicaciones a testear al centro de testing y desean ver los resultados de los tests ejecutados ya que poseen los passwords de los tests que se ejecutaron para sus productos. Las herramientas 1, 2 y 3 son las herramientas de testing de Performance que se encuentran incorporadas en PC ya que fueron hechos plugins para cada una de ellas. Las mismas ejecutan los tests que PC indica en las aplicaciones a testear: Aplicación 1, 2 y 3.

4.4 ARQUITECTURA

En esta sección se presenta un resumen de la arquitectura de PC. Se expone la arquitectura en capas que el mismo presenta, así como el diseño que permite fácilmente la implementación de nuevos plugins para hacer posible la interacción con nuevas herramientas de Testing de Performance. Por más información acerca de la arquitectura se puede consultar el Documento de Diseño del proyecto.

El sistema PC esta desarrollado siguiendo un proceso de desarrollo basado en casos de uso y centrado en la arquitectura. Posee un diseño en capas y prevé el agregado de nuevos plugins. Su interfaz es Web y utiliza como medio de persistencia una base de datos relacional.

La extensibilidad mediante el uso de plugins es otro aspecto muy importante del sistema desarrollado. Cada plugin que se agrega al sistema, le permite a éste, interactuar con una nueva herramienta de testeo, es decir, poder enviar tests a ejecutar a la misma y luego obtener los resultados de la corrida del test. El resultado logrado, permite a cualquier usuario programador implementar un plugin y agregarlo a la herramienta sin ningún tipo de dificultad. Para lograr este tipo de extensibilidad se generó un componente Adaptador que es el encargado de la extensibilidad del sistema por medio de plugins.

La organización de la arquitectura en capas implica que una capa puede utilizar servicios de otra inferior, pero no más de un nivel por debajo de ella. Las capas que lo conforman son las siguientes:

- Interfaz de Usuario
- Diálogos de Usuario
- Servicios de Sistema
- Servicios de Negocio
- Infraestructura
- Excepciones
- Datatypes

En el siguiente diagrama se puede observar como es la comunicación entre capas.

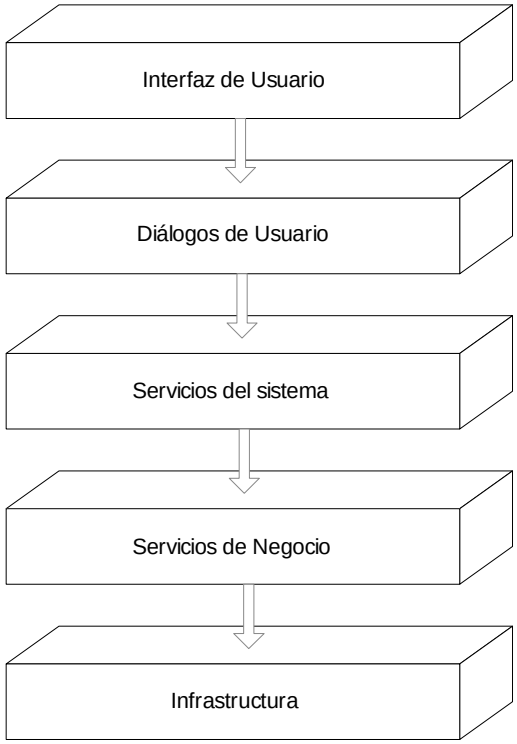

Figura 5: Diagrama de la estructura interna de Performance Center

A su vez, la capa de Servicios del Sistema está compuesta por 2 subcapas:

- Servicios del Sistema Locales
- Webservices

Y la capa de Infraestructura se descompone en:

- Plugins
- Data Mapper

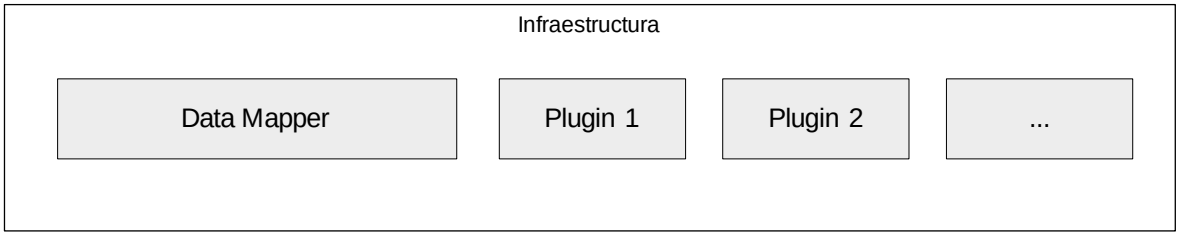

Figura 6: Capa de Infraestructura

A continuación se realiza un breve resumen de la función de cada capa explicando para cada una sus principales funciones.

- **Interfaz de Usuario:** La capa de de interfaz de usuario está compuesta por páginas Web dinámicas que permiten la interacción usuario - sistema de los distintos casos de uso.
- **Diálogos de Usuario:** La capa de diálogos de usuario contiene los módulos que llevan a cabo la lógica de los casos de uso. Mediante está lógica, cada módulo regula la secuencia de ventanas de la interacción arriba mencionada e invoca los servicios del sistema requeridos. En esta capa se implementan los gráficos que se despliegan en la interfaz de usuario.
- **Servicios de Sistema:** Esta capa representa la entrada a la lógica del sistema. Existen uno o más módulos por cada uno de los subsistemas identificados. Estos módulos ofrecen las interfaces requeridas por cada caso de uso asociado al subsistema y los controladores que las implementan. A su vez, la entrada a esta capa esta diseñada siguiendo el Design Pattern Factory.
- **Servicios de Negocios:** En esta capa, se encuentran los manejadores de los conceptos principales del modelo de dominio del problema así como todas las clases necesarias para que el sistema funcione. Estos manejadores encapsulan el acceso a los datos, ocultando para los servicios de nivel superior la forma en que los mismos son almacenados. Al igual que en la capa de Servicios de Sistema, la entrada a esta capa esta diseñada siguiendo el Design Pattern Factory. A diferencia de los módulos en las capas superiores los aquí presentes podrán ser compartidos por varios subsistemas. En esta capa también se provee una adaptación estándar e independiente con los plugins, cuya función es explicada mas adelante.
- **Data Mapper:** Capa en la cual se ubica la lógica de la persistencia, o sea la comunicación con la base de datos relacional.

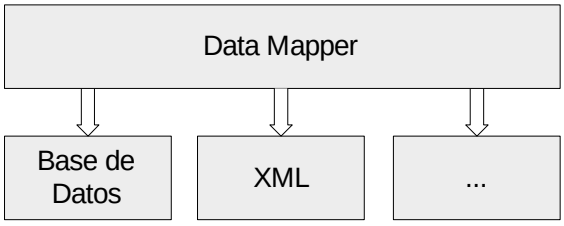

Figura 7: Data Mapper

La subcapa Data Mapper se comunica directamente con la Base de Datos y con estructuras XML que también son parte de la persistencia del sistema.

- **Plugins:** Esta capa ofrece los servicios de las distintas herramientas de performace que el sistema integra. La misma esta diseñada para que cualquier programador agregue plugins sin mayores dificultades permitiendo así, la integración de todas las herramientas de testing que se desee. Un Plugin es un modulo de software que puede ser escrito por usuarios

programadores y que se integra a la aplicación de forma automática, permitiendo que Performance Center pueda ejecutar tests y obtener resultados de nuevas herramientas.

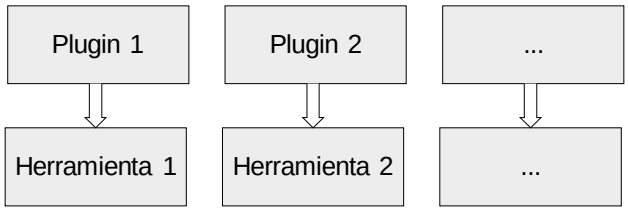

Figura 8: Plugins

Cada plugin se comunica con una herramienta de Testing de Performance diferente y permite a Performance Center el uso de la misma.

- **Webservices:** Esta capa es la encarga de implementar los servicios Web del sistema. La misma es una entrada independiente al sistema, implementa la lógica de las consultas y se comunica directamente con la capa de servicios de negocio. Para su implementación se contó con herramientas de alto nivel como será explicado mas adelante.
- **Excepciones:** Esta capa reúne todas las excepciones que el sistema utiliza y es utilizada por todas las capas mencionadas.
- **Datatypes:** Esta capa reúne todos los datatypes que el sistema utiliza para comunicar diferentes estructuras de datos entre capas y es utilizada por todas las capas mencionadas.

Las capas Excepciones y Datatypes son ortogonales al sistema y son utilizadas por todas las capas restantes.

Figura 9: Excepciones y Datatypes

4.5 DESIGN PATTERNS

Performance Center es un sistema Cliente/Servidor que se estructura según el patrón "Layers". Luego, en cada capa, se utilizaron diferentes patrones de diseño como ser: Model View Controller, Abstract Factory, Singleton y Data Mapper.

El patrón Model View Controller (MVC) es utilizado en la propia división en capas. El "Model" exporta las funcionalidades del sistema y está formado por la capa de Servicios del Sistema y todas las capas por debajo de ella. El "Controller" guarda el estado de los usuarios y está formado por los Servlets de Java agrupados en la capa de Diálogos de Usuario. Los Servlets están implementados siguiendo el patrón Front Controller. El "View" despliega la información al usuario y está formado por las páginas JSP. Las páginas JSP responden al patrón Template View.

El patrón Abstract Factory fue utilizado en los puntos de acceso a las capas de Servicios del Sistema y Servicios de Negocio. De esta forma, toda vez que se accede a las demás interfaces de estas capas, se hace a través de una única interfaz.

El patrón Singleton se utilizó en los manejadores de la capa de Servicios de Negocio que son los que mantienen las colecciones de los demás objetos.

Data Mapper es un patrón en el que un conjunto de mappers mueven datos entre objetos y la base de datos manteniéndolos independientes entre sí y con respecto al mapper. El mismo fue utilizado para independizar el modelo de dominio de la base de datos.

4.6 TECNOLOGÍAS UTILIZADAS

- **Java:** Se decidió utilizar como lenguaje de programación *Java*, debido a que se trata de un leguaje orientado a objetos, multiplataforma, de uso libre y con el cual el equipo de desarrollo ya poseía experiencia. Otra de las razones de elección de Java fue su amplio uso y la enorme cantidad de componentes existentes que pueden ser reutilizados y así poder enriquecer el resultado final de producto. La versión de java utilizada es la 1.4.2 [13].

- **Castor:** Para unir el diseño de objetos realizado y el modelo relacional se utilizó Castor, el cual es un framework Open Source para el "binding" de objetos a tablas relacionales y documentos XML. Con *Castor* se logra independizar la capa lógica de la aplicación, de la base de datos a utilizar y de la correspondencia con el modelo de datos. Por otra parte, condiciona relativamente el diseño de los objetos: la restricción más importante es sobre los identificadores de los objetos y las relaciones entre los objetos que deben ser en los dos sentidos. Se utiliza Castor para persistir los objetos a la base de datos y para recuperar los objetos se utilizan sentencias OQL. La transformación de sentencias de OQL a SQL se realiza de forma automática, es decir, en ningún momento se accede a la base de datos con una consulta SQL. *Castor* permite fácilmente trasformar objetos a XML con poco esfuerzo, lo cual fue de gran ayuda a la hora de cumplir el requerimiento de exportación e importación de Tests y Resultados. La versión utilizada fue la 0.9.5.3 que era la última versión estable disponible al comienzo de la implementación de PC. Cuando fue liberada la versión 0.9.6 en febrero del 2005 se sustituyó la versión anterior ya que habían bugs en la versión usada de Castor que estaban repercutiendo en el correcto funcionamiento de PC [14].

- **MySQL:** Luego de haber definido el uso de Castor, se eligió MySQL como manejador de bases de datos ya que es Open Source, de muy amplio uso y uno de los manejadores con los que Castor ofrece compatibilidad. MySQL cumplió con todos los requisitos a pesar de brindar solo los servicios básicos. Además resultó adecuado para un sistema de mediano porte como PC y ofrece herramientas para la Administración de la base de datos que son fáciles de usar. La versión utilizada de MySQL es la 4.1 [15].

- **Apache Tomcat:** Utilizamos el servidor Web Apache Tomcat 5.0 debido a su conocida facilidad de uso, su amplia difusión y la experiencia que poseía el equipo utilizando el mismo [16].

- **SNMP:** En el estudio de diversas arquitecturas para monitoreo de plataformas se encontró que una de las formas más estándares de recolectar información de las distintas arquitecturas era mediante consultas SNMP. Parar eso se utilizó un componente gratuito de AdventNet que permite realizar consultas SNMP de una forma sencilla y general. De esta manera, mientras Performance Center, envía a ejecutar un test a una herramienta, puede registrar mediciones SNMP de los hosts que desee para así monitorear su desempeño y tener más información para

poder evaluar de una forma certera el desenvolvimiento de los sistemas que están siendo puestos bajo prueba. La versión utilizada fue AdventNet SNMP API 4 [17].

- **Axis:** Esencialmente Axis (Apache EXtensible Interaction System) es un motor de SOAP o sea un framework para construir procesadores de SOAP. Está desarrollado en Java y se considera como la tercera generación de SOAP. Permite mediante archivos de definición XML y clases java brindar servicios Web. Es utilizado en su versión 1.1 para hacer posible de una manera fácil del desarrollo de webservices [18].

- **SHA:** Para encriptar las contraseñas de los usuarios y de los tests se utilizó Secure Hash Algorithm. De esta manera las contraseñas son guardadas de forma encriptada en la base de datos, lo que provee seguridad para el usuario.

- **JFreeChart:** Para facilitar el análisis y visualización de los resultados se decidió agregar gráficas. Este componente es sumamente configurable y permite generar gráficas de barras, puntos y de tortas. Se utilizó la versión 0.9.21 de JFreeChart, la cual permite con poco esfuerzo generar gráficos de diferentes tipos y de gran calidad [19].

- **Applet:** Para la visualización de directorios, ensayos, tests y resultados en la estructura de árbol, se utilizó un applet de Java. Este componente permite mover los Directorios, Ensayos y Tests utilizado la técnica drag-and-drop, y brinda una interfaz visual más agradable que una solución implementada mediante paginas JSP [20].

- **Eclipse:** Como entorno de desarrollo se utilizó Eclipse 3.0 lo cual facilitó la implementación por la alta cantidad de plugins que existen para el mismo, siendo el Plugin de Sysdeo [21] de Apache Tomcat el más importante. El mismo facilitó la tarea de desarrollo debido a la posibilidad de realizar debug del código de una forma sumamente sencilla y natural, al mismo tiempo que incorpora facilidades para generar la estructura necesaria de los proyectos Web. Durante el transcurso del proyecto se utilizó un repositorio de código fuente (CVS). Su utilización fue importante a la hora de programar en paralelo y de tener copias del código en servidores remotos. Eclipse trae incorporado herramientas para el manejo de CVS que facilitan el uso del repositorio y el trabajo concurrente de un equipo de desarrolladores [22].

CAPÍTULO 5: NIVEL DE CALIDAD

5.1 INTRODUCCIÓN

A todos los proyecto del área de Concepción de Sistemas de Información (CSI), que comenzaron en el año 2004, se les exigió que tuvieran un nivel de calidad mínimo. Los niveles de calidad exigidos fueron elaborados por el Ing. Diego Vallespir como parte de su tesis de maestría. Se definió una escala con ocho niveles de calidad, donde el nivel de calidad uno es el menos exigente y el nivel ocho es el más exigente. Al proyecto aquí presentado se le exigió un nivel de calidad tres.

A continuación se describe resumidamente el nivel de calidad tres, luego se presentan los resultados obtenido en el proceso de verificación y finalmente se presentan algunas métricas obtenidas. Todos los detalles referentes a los niveles de calidad, incluyendo el nivel tres, se encuentran detallados en [3].

5.2 NIVEL DE CALIDAD 3:

Cumpliendo las actividades de este nivel deberían conseguirse los siguientes objetivos entre otros:
- Conocer las fallas del sistema y poderlas reproducir.
- Conocer la cantidad de fallas relacionada con la cantidad de casos de prueba funcionales ejecutados.
- Conocer las funcionalidades del mismo que han sido cubiertas por los casos de prueba.

Entre las actividades definidas para este nivel se encuentran:
- Relevar Requerimientos y Priorizarlos
- Validar Requerimientos
- Generar Casos de Uso y Priorizarlos
- Validar Casos de Uso
- Testing de los Casos de Uso
- Testing Funcional del Sistema
- Test de Aceptación
- Establecimiento de Cubrimiento

5.3 RESULTADOS OBTENIDOS:

El proyecto se realizo siguiendo un proceso iterativo incremental basado en casos de uso y centrado en la arquitectura. Esto significa que se siguieron una serie de pasos como la elaboración de una arquitectura candidata y su posterior refinamiento que llevaron a una arquitectura estable y definitiva. Se puede establecer que los casos de uso condujeron la arquitectura porque la misma está influenciada por los casos de uso relevantes, y que la arquitectura guió los casos de uso porque los casos de uso a considerar y su realización dependieron de la arquitectura definida. Se realizaron tres iteraciones en las cuales al final de cada una se obtuvo un producto. A cada uno de estos productos se le realizó una verificación completa incluyendo pruebas de regresión. Durante el proyecto se planearon las actividades en base a los casos de uso, los cuales fueron implementados según el riesgo que representaban.

Dentro de las actividades especificadas en el nivel de calidad las primeras en realizarse fueron: Relevar Requerimientos, Generar Casos de Uso y la priorización tanto de Casos de Uso como de Requerimientos. La salida de estas actividades fueron el Documento de Requerimientos, el Documento de Casos de Uso y el Documento de Trazabilidad. Luego de cada cambio en los documentos se realizo la respectiva validación con el cliente.

Para registrar el cubrimiento de la especificación de requerimientos se elaboró el documento Casos De Prueba de Requerimientos en el cual se detalla un conjunto de pruebas para cada requerimiento. Todas las pruebas del documento se realizaron de forma exitosa.

Para realizar el testing de los Casos de uso y registrar el cubrimiento de los mismos se realizó el Plan de Pruebas del Sistema en el que se detallan los escenarios condición para cada caso de uso. El diseño de los casos de prueba fue hecho antes de comenzar con la codificación del cada uno y la realización de las pruebas fue hecha luego de terminada cada iteración. En las tres iteraciones fueron probadas todas las funcionalidades, es decir se hicieron pruebas de regresión. Los resultados de dichas pruebas se encuentran registrados en el Reporte de Fallas del Sistema.

Las pruebas de aceptación fueron hechas en base a un subconjunto de casos de prueba del sistema agrupados en ciclos. Los diferentes ciclos intentan reproducir usos típicos del sistema al mismo tiempo que abarcan todas las funcionalidades que el sistema provee. Las pruebas de aceptación fueron corridas junto al cliente al finalizar cada iteración, en ninguna de ellas se encontraron fallas.

Otras actividades relacionadas con el proceso de verificación fueron el establecimiento de cubrimientos (Plan de Verificación y Validación), la elaboración de un manual de usuario y el desarrolló de clases JUnit.

Luego de culminar la verificación de la tercera iteración se decidió corregir los errores más importantes y entregar un producto, junto al verificado, en versión Beta, es decir que no se verificó.

Para la gestión del proyecto se realizaron sucesivas planificaciones, se definieron actividades, se controlaron sus ejecuciones y se evaluaron los resultados. Como parte de la planificación del proyecto se realizaron varios cronogramas utilizando técnicas como la de los diagramas de Gantt. Anexado a este documento se pude encontrar el Plan del Proyecto correspondiente a la última etapa.

La etapa inicial del proyecto correspondió a la evaluación de herramientas y al estudio del estado del arte, su duración fue aproximadamente 120 días. En la siguiente etapa se relevaron requerimientos y se realizaron diversos prototipos, su duración fue aproximadamente 75 días.

La última etapa fue la implementación del producto y su planificación consistió en tres iteraciones de periodos de tiempos y cargas similares. Luego de realizadas las iteraciones se observaron algunas desviaciones que no tuvieron un impacto importante en el proyecto. A continuación presentaremos una tabla que resume los tiempos planificados y los reales.

Iteración	Cantidad de Casos de Uso	Cantidad de días planeados	Cantidad de días reales
1	3	31	40
2	6	29	29
3	6	28	42

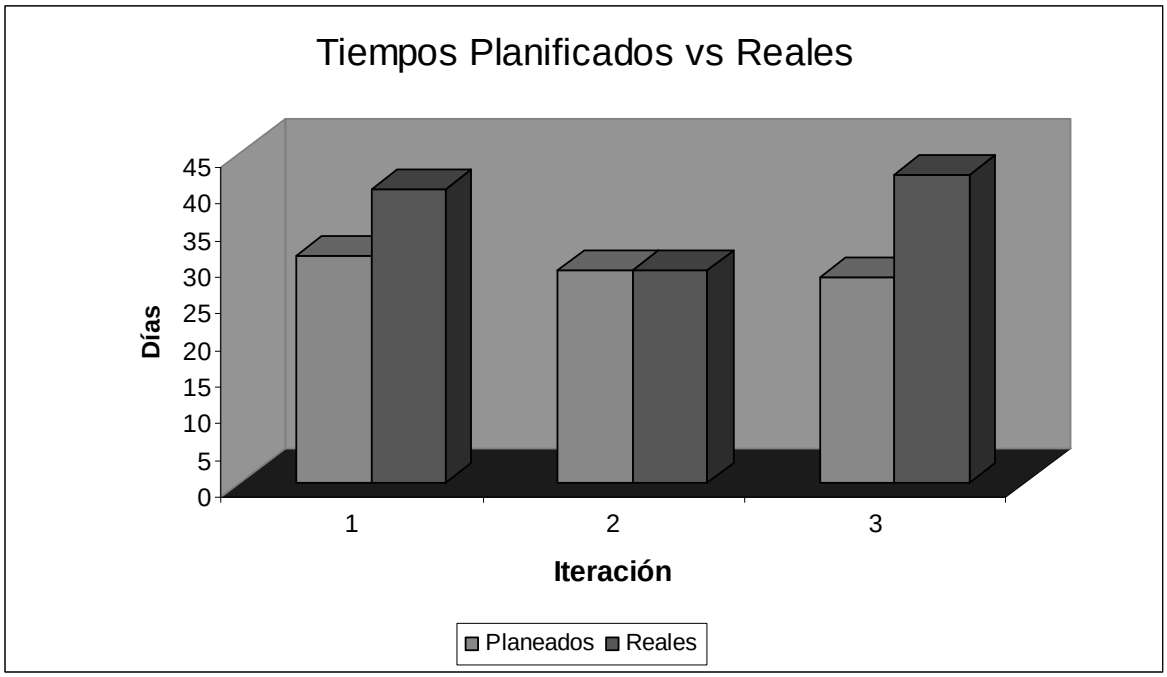

Figura 10: Tiempos Planificados vs. Reales

5.4 MÉTRICAS:

Aquí se presentan las dos métricas obtenidas del proceso de verificación, la primera es la cantidad de fallas sobre cantidad de casos de prueba, la segunda es la cantidad de fallas sobre la cantidad de líneas de código. A continuación presentamos los resultados y luego brindamos una interpretación de los mismos.

Iteración	Cantidad de Fallas	Cantidad de Pruebas	Cociente
1	10	49	0,204
2	20	108	0,185
3	13	180	0,072

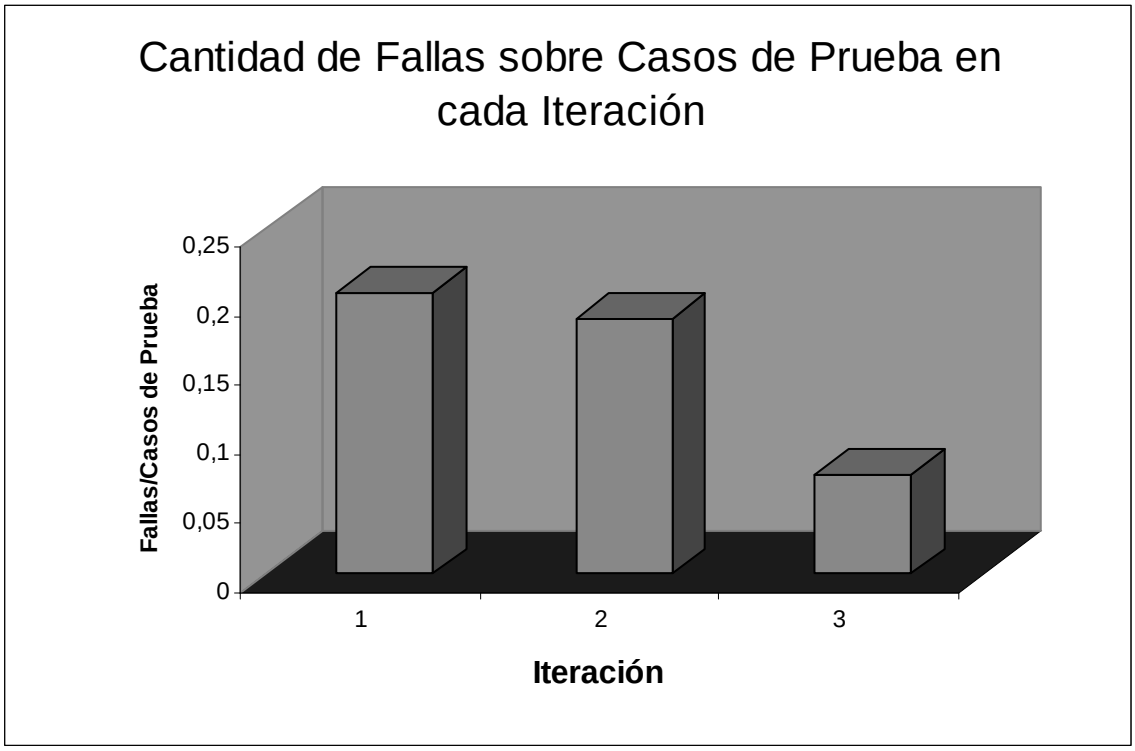

Figura 11: Cantidad de Fallas sobre Casos de Prueba

Iteración	Cantidad de Fallas	Líneas de Código	Cociente
1	10	4506	0,00221
2	20	5178	0,00386
3	13	7429	0,00174

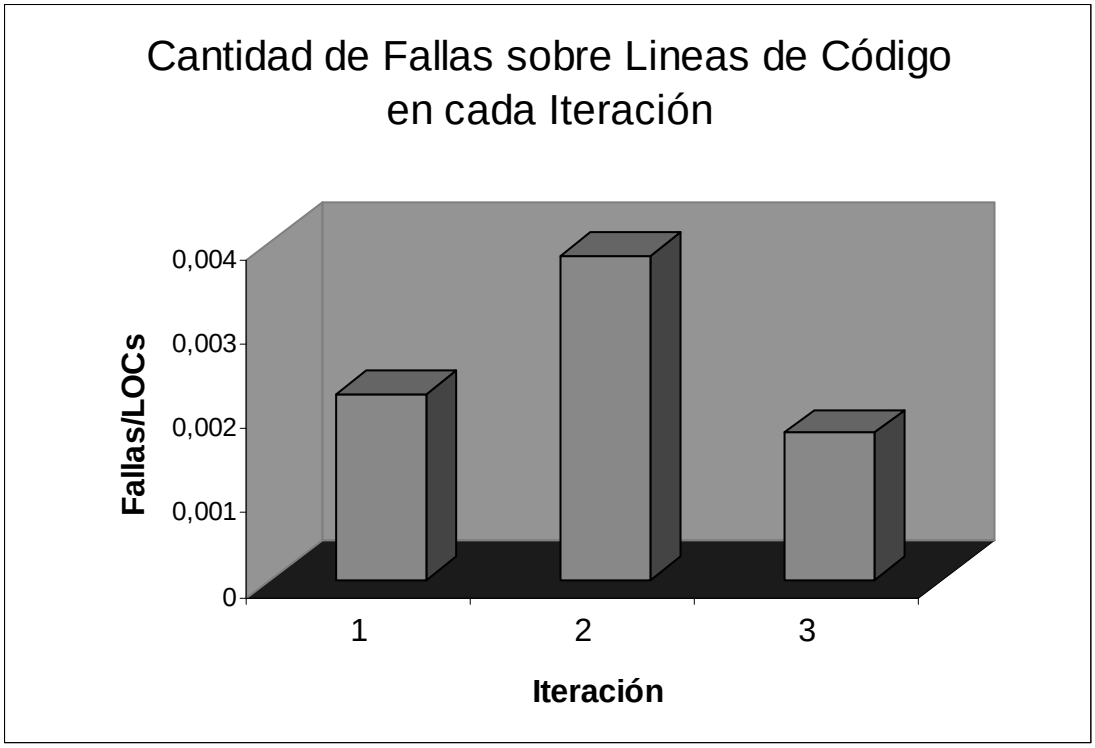

Figura 12: Cantidad de Fallas sobre Líneas de Código

La primera iteración abarcó pocas funcionalidades por lo cual la cantidad de pruebas fue mucho menor que en las demás iteraciones. Por ser pocas funcionalidades la cantidad de fallas que ellas presentaron fue baja. Esta iteración fue en la que se codificaron más líneas de código, esto fue debido a que se reutilizó código de pequeños prototipos realizados anteriormente.

En la segunda iteración se agregaron varias funcionalidades por lo cual aumentó la cantidad de Pruebas. La cantidad de fallas aumentó de forma importante en esta iteración ya que por estar en pleno desarrollo se dejaron algunos detalles para resolver en la iteración siguiente, la mayoría de estás fallas eran de incidencia baja. La cantidad de líneas de código no aumentó de forma importante por tratarse de una iteración corta y porque no se reutilizaron módulos previamente implementados.

En la última iteración se agregaron más funcionalidades y se intentó obtener un producto con mejor calidad que en las iteraciones anteriores, por lo que se aumentó la cantidad de casos de prueba de forma considerable. La cantidad de fallas encontradas respondió a las expectativas y el cociente descendió de forma importante. La cantidad de líneas de código producidas fue mayor que en la segunda iteración por tratarse de una iteración más larga.

CAPÍTULO 6: CONCLUSIONES Y TRABAJOS FUTUROS

6.1 CONCLUSIONES

En está sección se realizara una breve evaluación del trabajo realizado y luego se presentarán algunas conclusiones de carácter general. Los objetivos del proyecto fueron: un estudio del estado del arte del Testing Performance, la evaluación de herramientas para "Testing de Performance" y la implementación de un Toolkit para Testing de Performance.

Como resultado del estudio del estado del arte del Testing Performance se realizó un artículo que se encuentra anexado a este documento. Durante la elaboración de dicho artículo se encontró que a pesar de la importancia que está tomando el área, hay poca bibliografía sobre los aspectos teóricos y metodológicos del Testing Performance. Incluso al tomar distintas fuentes se puede observar que no existe consenso en algunas definiciones básicas. Por está razón el artículo elaborado de Testing de Performance es un aporte muy importante de este proyecto. En el artículo se brinda la definición de los conceptos más importantes recabados de distintas fuentes y se presenta una lista con las referencias más importantes dentro del área. La evaluación de está parte del proyecto es muy positiva porque todo lo aprendido ayudó al desarrollo del resto del proyecto.

El segundo objetivo del proyecto fue la evaluación de herramientas para "Testing de Performance" que tuvo como resultado la Tabla Comparativa que se encuentra anexada al final del documento. Entre los puntos positivos de este trabajo se encuentran la metodología seguida, la cantidad de herramientas evaluadas y el conocimiento adquirido. Haber seguido una metodología estructurada fue bueno para ordenar esta etapa del proyecto, la cantidad de herramientas evaluadas es respetable considerando la cantidad de herramientas existentes en el mercado, y el conocimiento adquirido sobre las funcionalidades fue decisivo para diseñar el producto final. Por otro lado entre las dificultades encontradas durante la evaluación está el no poder instalar los productos comerciales debido a que no se disponía de los paquetes de instalación, el hardware apropiado y de las licencias correspondientes. Por esta razón, la evaluación de estas herramientas se basó en la información publicada por el fabricante y en algunos artículos independientes. Una segunda dificultad fueron los problemas en la instalación de algunas herramientas open source que imposibilitaron su uso. Otra dificultad fue que las páginas de algunos productos tenían información escasa o desactualizada. El aporte más importante de esta parte del proyecto es una Tabla Comparativa que resume las características de todas las herramientas evaluadas. La evaluación de esta etapa es positiva pero como crítica se puede decir que la evaluación podría haber sido más profunda aún. Por ejemplo, se podría haber evaluado una misma aplicación de mediano porte con las distintas herramientas. La razón por la cual no se hizo este tipo de evaluación fue debido a que para realizarlo sería necesario más tiempo, más conocimiento en el uso de cada herramienta y más recursos de hardware. Los recursos de hardware podrían haber sido facilitados por el CES, pero lamentablemente esto no ocurrió debido a que el Laboratorio de Ensayos no contó con equipos durante el 2004.

El tercer objetivo del proyecto fue la implementación de un Toolkit para Testing de Performance que permitiera realizar los tests de forma eficiente, eficaz y simple. El conjunto de herramientas (Toolkit) propuesto está formado por aplicaciones comerciales existentes y la aplicación "Performance Center". Para ayudar a seleccionar las aplicaciones existentes se realizó la evaluación de herramientas y para integrarlas a todas ellas se desarrolló el producto "Performance Center".

La principal dificultad en su desarrollo fueron los problemas relacionados con los componentes utilizados, en particular con Castor. Como ejemplo se puede mencionar que durante la

implementación se encontraron problemas en la aplicación debido a errores de Castor que se solucionaron actualizando la versión. A pesar de los problemas encontrados se recomienda su uso porque la transformación entre el modelo de objetos y el modelo relacional que se logra es muy prolija, y el grado de independencia con la base de datos es total. Otra dificultad presente a lo largo de todo el proyecto fue la imposibilidad de estimar el tiempo que llevaría el aprendizaje de cada tecnología utilizada. Las curvas de aprendizaje durante el proyecto fueron muchas, debido a la cantidad de componentes utilizados y a la falta de experiencia del equipo de desarrollo en ellos. Entre estas tecnologías se encuentra: Castor, Eclipse, Axis (Web Services), SNMP, Applets, JSP, Servlets, JFreeChart.

Entre los puntos que se plantearon al inicio del proyecto pero que luego no se realizaron se encuentra la definición de un script genérico para todas las herramientas. Este punto es muy ambicioso debido a la diversidad de lenguajes que utilizan las herramientas y la propia naturaleza de los Tests de Performance, en los cuales los sistemas a testear pueden presentar características totalmente distintas. Por otro lado el punto en el cual siempre se va poder evolucionar y mejorar es en la definición de la interfaz por la cual "Performance Center" interactúa con las otras herramientas. Para realizar un mejor trabajo en este punto sería bueno contar con retroalimentación por parte de usuarios, para tener información por ejemplo sobre que otras medidas les interesaría incluir. En la sección de trabajos futuros se plantean varios aspectos a mejorar de "Performance Center".

La evaluación del producto es satisfactoria aunque como todo producto de software siempre puede ser ampliado y mejorado para brindar más servicios y de mejor calidad. En resumen se puede decir que el producto presenta una idea novedosa y muy útil para el "Testing de Performance".

Sobre la gestión del proyecto se puede destacar que la planificación, coordinación y control de las actividades del proyecto fue simple y no presento problemas, debido a que el grupo estuvo integrado por solo dos personas. A pesar de no haber hecho un control estricto de las horas trabajadas se puede concluir que estas superaron ampliamente la carga horaria establecida para los Proyectos de Grado, no solo por la cantidad de horas dedicadas por semana sino también por la duración de todo el proyecto.

Para finalizar se puede concluir que el Test de Performance es importante a la hora de evaluar el éxito de un producto ya que no es fácil hallar usuarios dispuestos a utilizar sistemas lentos en los que las tareas insuman tiempos considerables por más que el producto funcione correctamente. Segundo que es importante que el conjunto de herramientas con el cual se va a realizar las pruebas sea cuidadosamente seleccionado, teniendo en cuenta sus capacidades para generar, desarrollar y reportar resultados de Tests de Performance sobre un software determinado.

6.2 TRABAJOS FUTUROS

Como líneas de trabajo futuro se proponen las siguientes posibles mejoras a "Performance Center":

- Implementar Plugins para una mayor cantidad de herramientas. De está forma el usuario dispondría de una cantidad importante de herramientas en las cuales ejecutar sus Tests de Performance.

- La posibilidad de configurar los distintos Plugins desde la propia herramienta. Por lo general los Plugins tienen parámetros configurables, por ejemplo, los que se adjuntan con la herramienta utilizan archivos XML para establecer dichos parámetros. Estos parámetros son propios de cada Plugin por lo que no existe un modelo común sobre el cual trabajar. Entonces se tendrían

que crear menús y formularios dinámicos que permitieran establecer los parámetros dentro de "Performance Center". Esta forma podría ser por medio de metadata y archivos XML.

- Proveer de un explorador de MIB's para la creación de consultas SNMP. Esto permitiría al usuario dar de alta consultas sin la necesidad de recurrir a exploradores externos para obtener los identificadores de los objetos de las mismas.

- Crear procesos agentes que permitan la comunicación con herramientas remotas. "Performance Center" solo se comunica con las herramientas de testing de forma local, por lo que la creación de estos agentes permitiría comunicarse con distintas herramientas distribuidas en una red. Con esto se lograría tener más posibilidades en la ejecución de los tests.

- Agregar funcionalidades para aumentar la asistencia en todos los requerimientos del Toolkit. Los requerimientos fueron explicados en el capitulo tres y son los siguientes: utilidades para crear y mantener los datos de pruebas, herramientas para generar carga al sistema, herramientas para ejecutar las transacciones, monitores de los recursos del sistema, utilidades para registrar y analizar los resultados obtenidos. Por ejemplo, una funcionalidad que podría agregarse es la capacidad de almacenar los datos de prueba de los tests dentro de "Performance Center".

- Graficas en tiempo real de los resultados SNMP y de los tests, para que el usuario pueda estar al tanto de la evolución de la ejecución de los mismos.

- Capacidad de realizar "On-Line Analytical Processing" (consultas OLAP), es decir convertirlo en un sistema de soporte a las decisiones que acceda a los datos de forma eficiente. Esto permitiría al usuario extraer y visualizar los datos desde distintos puntos de vista de forma simple.

- Capacidad para realizar predicciones de resultados, esto podría ser muy útil por ejemplo para determinar puntos de quiebre en el comportamiento del software. Las predicciones podrían realizarse mediante modelos estadísticos y usando técnicas como el de regresión lineal.

- Definir un lenguaje universal para los scripts de los tests, con esto se lograría que los mismos dejen de pertenecer a una herramienta específica. Para esto debería crearse un nuevo lenguaje o tomar alguno ya existente como lenguaje estándar de "Performance Center". Para la ejecución un test en cada herramienta se debería implementar un traductor que transforme el lenguaje de "Performance Center" en el lenguaje de la herramienta en la que se quiere correr el test.

REFERENCIAS

[1] Concepción de Sistemas de Información, Instituto de Computación, Facultad de Ingeniería, Universidad de la República,
http://www.fing.edu.uy/inco/grupos/csi/
Última visita: 10 de Marzo de 2005

[2] Jorge Triñanes, Centro de Ensayos de Software UDELAR, Facultad de Ingeniería, Instituto de Computación, Montevideo, Uruguay.
JIISIC'04 - Noviembre de 2004

[3] Diego Vallespir, Tesis de maestría (en curso), Actividades para mejorar la calidad Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo Uruguay.
http://www.fing.edu.uy/~dvallesp/Tesis/webActividades/index.htm
Última visita: 21 de Noviembre de 2004

[4] Centro de Ensayos de Software,
http://www.ces.com.uy/
Última visita: 10 de Marzo de 2005

[5] Paul Gerrard and Andrew O'Brien, "Client/Server Performance Testing", EuroSTAR '95, London UK, 27-30 November 1995.

[6] FAQ del grupo comp.software.testing,
http://www.faqs.org/faqs/software-eng/testing-faq/
Última visita: 10 de Marzo de 2005

[7] Grupo comp.software.testing:
http://groups.google.com.uy/groups?hl=es&lr=&selm=20000320182100.00937.00000062%40ng-bd1.aol.com
Última visita: 10 de Marzo de 2005

[8] Paul Gerrard, Neil Thompson, "Risk Based E-Business Testing", Chapter 12: Service Testing, 2002.

[9] An overview of load test tools, Julien Buret, Nicolas Droze, 2003,
http://clif.forge.objectweb.org/
Última visita: 10 de Marzo de 2005

[10] Abraham Jacob, Riyaj Shaik, Paul Tennis, "Load Test Tools Evaluation", 2002.
http://www.grove.co.uk/
Última visita: 10 de Marzo de 2005

[11] Open Source Testing,
http://www.opensourcetesting.org/performance.php
Última visita: 10 de Marzo de 2005

[12] Load and Performance Tools,
http://testingfaqs.org/t-load.html
Última visita: 10 de Marzo de 2005

[13] Java 2 Platform, Standard Edition (J2SE) Version 1.4.2,
http://java.sun.com/j2se/1.4.2/
Última visita: 10 de Marzo de 2005

[14] The Castor Project,
http://www.castor.org/
Última visita: 10 de Marzo de 2005

[15] MySQL,
http://www.mysql.com/
Última visita: 10 de Marzo de 2005

[16] Apache Jakarta Tomcat,
http://jakarta.apache.org/tomcat/
Última visita: 10 de Marzo de 2005

[17] AdventNet SNMP API 4,
http://snmp.adventnet.com/
Última visita: 10 de Marzo de 2005

[18] Apache Web Services Project,
http://ws.apache.org/axis/
Última visita: 10 de Marzo de 2005

[19] JFreeChart,
http://www.jfree.org/jfreechart/
Última visita: 10 de Marzo de 2005

[20] Java Applets,
http://java.sun.com/applets/
Última visita: 10 de Marzo de 2005

[21] Sysdeo Tomcat Plugin,
http://www.sysdeo.com/eclipse/tomcatPlugin.html
Última visita: 10 de Marzo de 2005

[22] The Eclipse Foundation,
http://www.eclipse.org/
Última visita: 10 de Marzo de 2005

[23] Jython Home Page
http://www.jython.org/
Última visita: 10 de Marzo de 2005