PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 08-20

Metodología para Pruebas de Desempeño

Federico Toledo **Matías Reina** **Simon de Uvarow**
Horacio López **Gustavo Vazquez** **Edgardo Greisin**

2008

Metodología para Pruebas de Desempeño
Toledo, Federico; Reina, Matías; Uvarow, Simon de; López, Horacio; Vazquez, Gustavo; Greisin, Edgardo
ISSN 0797-6410
Reporte Técnico RT 08-20
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, 2008

Metodología para Pruebas de Desempeño

Federico Toledo, Matías Reina, Simon de Uvarow, Horacio López, Gustavo Vazquez, Edgardo Greising

Centro de Ensayos de Software,
Universidad de la República,
Instituto de Computación,
Montevideo, Uruguay

{ftoledo, mreina, sduvarow, hlopez, gvazquez, egreising}@ces.com.uy

Resumen. Dentro de los servicios que brinda el Centro de Ensayos de Software (CES) se encuentran las pruebas de desempeño de productos de software desarrollados por terceros. Uno de los desafíos de estas pruebas es lograr reducir riesgos del negocio y obtener información útil del sistema en tiempos reducidos, tratando de obtener información de valor que permita maximizar la relación costo/beneficio de la prueba. Para poder atacar la complejidad que presentan este tipo de pruebas sin perder el foco en sus metas, es necesaria una metodología aplicable que provea un mapa de ruta en el proyecto de pruebas de desempeño. Este artículo presenta una metodología desarrollada por el CES y utilizada con éxito en proyectos críticos de la industria, con fundamento académico, pero sobre todo sumamente práctica.

Palabras Clave: Ingeniería de Software, Sistemas de Información

1 Introducción

Una prueba de desempeño (*performance test*) se define como una investigación técnica para determinar o validar la velocidad, escalabilidad y/o características de estabilidad de un sistema bajo prueba [1]. Las pruebas de carga (*load test*) tienen como objetivo simular la realidad a la cual estará sometido el sistema en producción (lo cual se conoce como escenario) para analizar su desempeño ante esa situación.

En este trabajo se presenta una metodología que se encuentra en el marco de la Ingeniería de Software más precisamente en el área de Verificación de Software, útil para realizar pruebas de carga, aunque puede ser extendida a otros tipos de pruebas de desempeño como por ejemplo pruebas de estrés o pruebas de picos. Las pruebas de estrés están enfocadas en determinar o validar las características de desempeño de un sistema expuesto a condiciones más allá de las anticipadas para producción mientras que las pruebas de picos se refiere a exponer al sistema a condiciones repentinas de incremento en la carga por un periodo corto de tiempo.

La metodología consta de seis etapas: Etapa inicial, Análisis de requerimientos, Automatización, Armado de la infraestructura definitiva, Ejecución y reportes y Etapa final. La metodología ha sido pensada para ser usada por una empresa de pruebas tercerizadas (*outsourcing*) como lo es el Centro de Ensayos de Software (CES) [3], por lo que hace foco en identificar claramente las responsabilidades de todas las partes involucradas y permite analizar qué actividades son convenientes realizar in situ con el cliente y cuáles se pueden realizar a distancia. Sin embargo, la misma puede ser utilizada tanto por la empresa de desarrollo para realizar sus pruebas de desempeño como por cualquier empresa que adquiere un nuevo sistema.

Otro punto fuerte de esta metodología es que no está atada a una herramienta particular. Toma algunas ideas de otras metodologías, por ejemplo de las ideas propuestas por Scott Barber en [10], pero generalizando la metodología a distintos tipos de aplicaciones, y a la utilización de cualquier herramienta de generación de carga. Existen algunas metodologías pero estas generalmente están atadas (o sugieren) la utilización de una determinada herramienta (por ejemplo [11]), y la metodología depende de esta. El CES necesita una metodología que se adapte a la herramienta que provea el cliente, o en su defecto, a la que se determine para el proyecto en que se aplique en particular.

Dicha metodología ha demostrado ser efectiva para la temprana detección de problemas de desempeño, ayudar a realizar la "sintonía" de los diferentes componentes del sistema y conocer los límites del sistema antes de alcanzarlos. Contar con una metodología probada para realizar pruebas de desempeño ayuda también a estimar el esfuerzo y los recursos necesarios para la realización de este tipo de pruebas.

En el presente trabajo se muestra su aplicación en un caso de estudio concreto. Se trata de un proyecto de prueba de desempeño realizado para una entidad bancaria previo a la migración de su sistema central. En este contexto se exigía la realización de pruebas de desempeño antes de la puesta en producción debido a que fallas en los tiempos de respuesta o en la estabilidad del sistema pueden ocasionar cuantiosas pérdidas a la institución. El proyecto de

pruebas resultó exitoso, y permitió a la entidad bancaria minimizar los riesgos de bajo desempeño y de inestabilidad del sistema al encontrar una serie importante de problemas y oportunidades de mejoras.

Es importante destacar que el área de estudio de pruebas de desempeño es reciente, las metodologías sobre estas pruebas no se encuentran ampliamente desarrolladas ni estandarizadas.

El documento se organiza de la siguiente manera: En la siguiente sección 2 se describe cada una de las etapas del proceso junto con sus actividades, objetivos, roles y entregables. En la sección 3 se describen las principales ventajas y desventajas encontradas al seguir la metodología sugerida en un caso de estudio particular. Por último se presentan las conclusiones y se plantea un posible conjunto de mejoras y extensiones al proceso como trabajo futuro.

2 Metodología para pruebas de desempeño

La complejidad de las tareas que se deben desarrollar para que una prueba de desempeño sea útil es tal, que merece el tratamiento de dicha actividad como un proyecto en sí mismo. En esta sección se presenta el proceso metodológico seguido para realizar pruebas de desempeño, que puede ser modificado y adaptado según el tipo de prueba de desempeño a realizar. En particular, este documento se enfoca en las pruebas de carga.

Se trata de un proceso en etapas secuenciales, con la particularidad de que la quinta etapa es iterativa. Estas etapas y sus principales actividades se ven en la *Fig. 1* y las mismas se describen en las subsecciones siguientes.

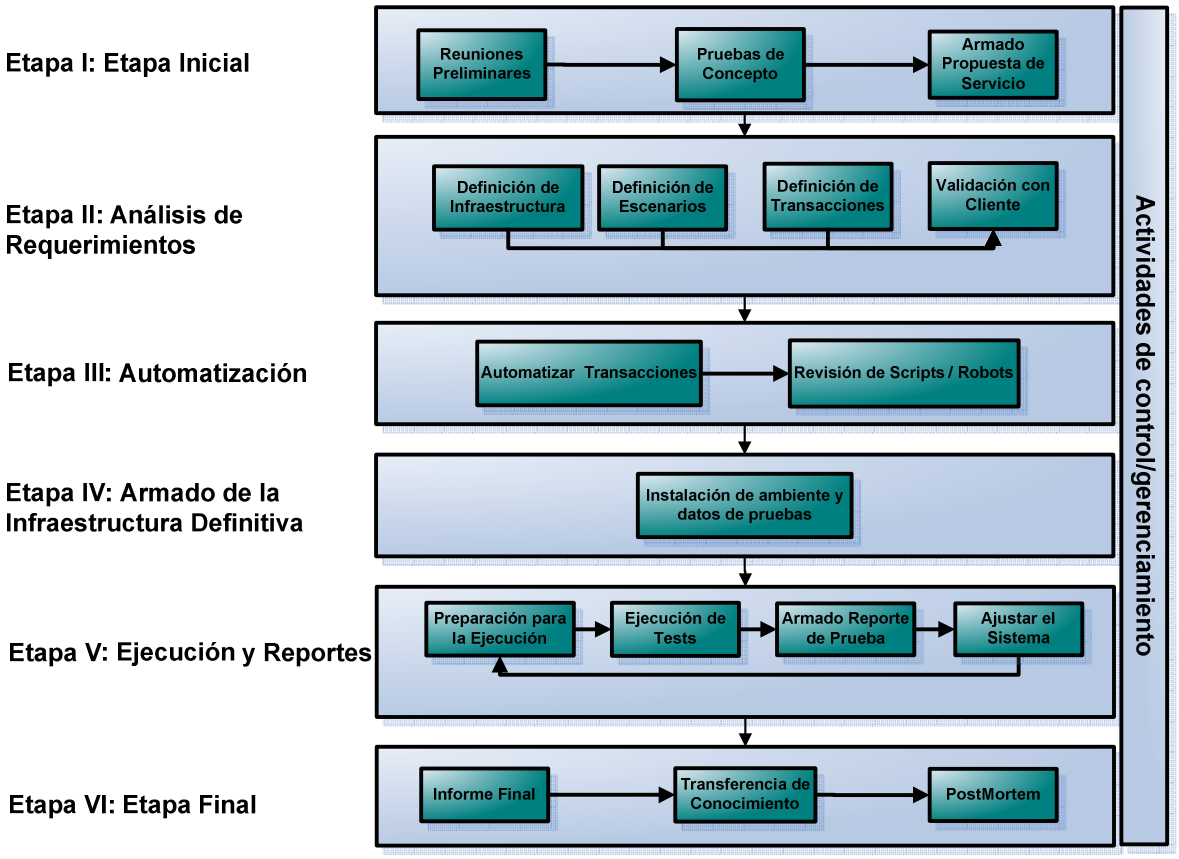

Fig. 1. Etapas y principales actividades de la metodología propuesta

La ejecución del proceso de pruebas de desempeño está a cargo de profesionales que cumplen *roles* bien definidos. Del lado del equipo de pruebas, existe un líder de proyecto quien se encarga de la gestión del proyecto, e interactúa con su contraparte en la empresa cliente, por lo general el gerente de tecnología o el líder de proyecto de desarrollo. Esta es la persona responsable del proyecto de pruebas por parte del cliente. Además, se cuenta con los "testers" que son los idóneos en las distintas herramientas de testing utilizadas en el proyecto, así como también del proceso. En contraparte al equipo de pruebas, se tienen varios roles. Además del líder de proyecto antes nombrado, existen los roles de experto funcional y experto de infraestructura. Los expertos funcionales son personas

conocedoras del dominio de la aplicación o del negocio, mientras que los expertos de la infraestructura son personas idóneas en los distintos componentes del sistema.

2.1 Etapa Inicial

Esta etapa tiene como objetivo definir el alcance global del proyecto de pruebas, por este motivo al finalizar esta etapa todas las partes involucradas deben conocer con detalle qué compromisos tiene cada una para que el proyecto se pueda realizar de manera exitosa.

En esta etapa se da el primer acercamiento entre las partes. Para definir el proyecto, se realizan reuniones en las cuales se presentan los principales conceptos relacionados a las pruebas de desempeño y se le entrega al cliente un cuestionario básico sobre las características de la aplicación a testear y el contexto bajo el cual se realizarán las pruebas.

A menudo para lograr una mejor estimación del esfuerzo, es necesario realizar *Pruebas de Concepto* en donde con el mínimo costo, se puedan analizar las particularidades del producto a probar para, entre los puntos más importantes, definir cuales son las herramientas más convenientes para las características particulares del proyecto. El objetivo de dichas pruebas es mitigar riesgos técnicos, y conocer la dificultad asociada a las distintas actividades del proyecto.

Luego de que se ha recopilado la información medular para definir el proyecto se procede a la elaboración de la propuesta de servicio. Dicha propuesta de servicio plasma en un único documento el contexto de las pruebas de desempeño y el alcance de las mismas, estableciendo de manera precisa los compromisos de cada parte involucrada. Entre otros puntos se precisan las personas involucradas y los roles que tendrán en el proyecto, un cronograma de pruebas, un presupuesto y un acuerdo de confidencialidad.

2.2 Análisis de Requerimientos

Esta etapa se comienza luego de concluida la etapa inicial, la cual provee una idea clara del contexto general del proyecto. El objetivo primordial de esta etapa es profundizar la recopilación de datos que se comenzó en la etapa inicial. Los principales tópicos a analizar y definir son los escenarios, las transacciones a incluir en las pruebas, la infraestructura sobre la cual correrá las mismas y los datos a utilizar.

Los *escenarios* especifican las diferentes condiciones de uso que debe soportar el sistema cuando el mismo se utilice en producción (este término es conocido como *workload* en la referencia [1]). El término transacciones se refiere en este contexto a los ciclos funcionales que pueden ejecutar los usuarios del sistema. Los escenarios son el punto más complejo de los requerimientos. Generalmente se asocia un escenario a un determinado momento del día, por ejemplo se puede definir un “escenario diurno” como la operativa que determina la carga que sufre el sistema a diario de 13:00 a 14:00 horas. Esa carga prevista se toma como referencia y se la denomina escenario del 100% y como se verá en la etapa *2.5 Ejecución y* Reportes, luego se ejecutan distintos porcentajes de ese modelo de carga.

Una vez definida la ventana horaria a simular en las pruebas se deben definir todos los componentes que conforman el escenario. Estos componentes son entre otros las transacciones que se ejecutan en esa ventana horaria, la cantidad de usuarios que ejecuta cada transacción, la cantidad de veces que ejecutan cada transacción, la forma de ingreso al sistema por parte de los usuarios y los procesos en lote que pueden correr en paralelo en dicho horario. Asociado a los escenarios se define la infraestructura sobre la que el sistema será puesto en producción. Es medular en las pruebas de desempeño poder definir estos datos de la manera más precisa posible, ya que luego la simulación de la carga se basará justamente en esta definición. Para poder analizar y definir todos los elementos del escenario, es imprescindible contar con un equipo interdisciplinario con responsables funcionales y desarrolladores así como también basarse en estadísticas, ya sean del mismo sistema que se va a implantar o de alguno anterior que se usara en la operativa a modelar.

Por otra parte, se debe definir un guión preciso para cada transacción donde se incluya cada acción que realiza el usuario para ejecutarla, y la correspondiente respuesta del sistema. Cabe destacar que se debe buscar mantener acotada la cantidad de transacciones que se incluirán en la prueba para poder manejar la complejidad que implica la automatización de las mismas. El criterio para la selección de las transacciones dentro del escenario elegido se basa en un análisis de riesgo, seleccionando aquellas que se ejecutan masivamente sobre el servidor, aquellas que se estima que realicen un consumo elevado de recursos o aquellas que son críticas para el negocio. Otro criterio utilizado para poder reducir la cantidad de transacciones es la simplificación de la realidad. Si tenemos, por ejemplo, una transacción de inserción y otra de modificación, tal vez podemos incluir en la prueba sólo una (la más crítica o riesgosa), y tomar de las estadísticas de ejecución la suma de ambas, considerando que el consumo de recursos de las dos es similar o equivalente. En este caso, se busca hacer la prueba realizable, aún cuando el modelo que se ejecute no sea 100% fidedigno con la realidad.

En cuanto a *la infraestructura* se debe precisar la versión de cada uno de sus componentes lógicos y físicos (software de base y hardware) así como también recopilar los indicadores de primer y segundo nivel a medir (los cuales se detallan en la sección *Infraestructura*). Es importante contar con técnicos responsables para la obtención de las métricas definidas para los diferentes componentes así como aquellos expertos que, en función de los resultados que se van obteniendo en las sucesivas ejecuciones, sean capaces de sugerir mejoras, realizando la "sintonía" o "*tunning*" de la configuración. Dentro de la infraestructura definida deben considerarse todos aquellos elementos que se vayan a ejecutar en producción concurrentemente con el escenario.
Otro punto importante es definir *los datos* que se utilizarán para ejecutar cada transacción del escenario de carga, ya que los mismos son una condición imprescindible para comenzar con la ejecución. Se definen tanto los datos de la base de datos, como los datos que se usarán como entrada para las transacciones a ejecutar. Es necesario que los datos sean similares a los que se espera tener en producción, tanto en calidad como en cantidad; de no ser así no se estaría modelando adecuadamente la realidad y muchos de los potenciales incidentes pueden no llegar a observarse en las pruebas. Estos datos pueden surgir de migraciones de sistemas anteriores, o pueden ser creados manual o automáticamente. Como se verá más adelante en la etapa de *3.4* Automatización, se pueden utilizar los artefactos creados en dicha etapa para asistir a la tarea de creación de datos. A menudo la creación de datos históricos, las pruebas de sistemas basados en flujos de trabajos (*workflow*) o la conversión de datos actuales a datos anónimos (para no exponer información de acceso restringido) pueden hacer de la actividad de creación de datos una actividad muy demandante de recursos. Estos recursos generalmente no son del equipo de pruebas, sino del equipo de desarrollo o funcional del cliente, debido, principalmente, al conocimiento que los mismos tienen del esquema de datos sobre el que se basa el sistema. La tarea de seleccionar los datos a utilizar como entrada de las transacciones es también crítica, ya que la variedad que entre ellos exista debe reproducir la de la realidad. Una mala mezcla de datos de entrada puede arruinar una prueba de desempeño.

Finalmente, es deseable establecer un criterio de aceptación u objetivo a superar. Este puede ser definido en base a una cantidad de operaciones a realizar en una ventana de tiempo, al consumo de recursos del sistema o al tiempo de respuesta de cada transacción, cualquiera de ellos medido en la situación de carga del sistema definida por el escenario correspondiente. Este criterio de aceptación es el principal objetivo a ser alcanzado y, ante todo, el principal elemento que define cuándo es aceptable dejar de ejecutar las pruebas.

Esta etapa se da por finalizada cuando se validan los datos definidos con todas las partes involucradas en el proyecto.

2.3 Automatización

La etapa de automatización tiene como objetivo construir los artefactos necesarios para poder reproducir de manera automática el escenario definido en la etapa *2.2 Análisis de* Requerimientos, esto generalmente incluye los scripts o robots que oficiarán de usuarios virtuales y los datos de entrada para los mismos. Esta automatización es requerida debido a que la alternativa de reproducir el escenario con usuarios reales frente a la aplicación tiene un costo muy elevado, tanto en equipamiento necesario como en recursos humanos que participen en la misma. La automatización, además, permite repetir las pruebas de manera sencilla en diferentes momentos, por lo que la repetición de las mismas no implica costos extra por el lado de la generación. De este modo, en caso de encontrarse una oportunidad de mejora o reparar un error, se puede realizar una prueba de la efectividad del mismo mediante la reejecución de las pruebas con exactamente los mismos datos y contexto. Esto no es posible de lograr, a costos razonables, de ninguna otra forma que no sea con pruebas automatizadas. A su vez, es la automatización lo que permite tomar una metodología de pruebas iterativas.

Los principales artefactos a construir son los scripts o robots capaces de ejecutar las transacciones, siguiendo el guión definido en la etapa anterior. Estos scripts, reproducidos por los denominados "usuarios virtuales", son capaces de ejecutar una transacción logrando que el servidor no distinga si la misma es ejecutada por un usuario real o un usuario virtual.

Para la producción de los scripts existen varias herramientas que permiten la creación de los mismos y luego la ejecución de manera eficiente de cientos de usuarios virtuales por maquina. Estás herramientas proveen la capacidad de ingresar los distintos parámetros del escenario definido en la etapa anterior, como son la cantidad de usuarios que ejecuta cada transacción, la cantidad de veces que ejecuta cada usuario, la forma de ingreso de los usuarios al sistema (cadencia), el tiempo en el que se ejecuta el escenario, etc. Además permiten la monitorización de la carga (por ejemplo la cantidad de usuarios activos) y de los tiempos de respuesta de cada transacción. Por otro lado permiten distribuir los usuarios virtuales en varias máquinas y algunas tienen incorporadas componentes de monitorización de infraestructura.

Muchas veces, debido a los distintos protocolos de comunicación que un sistema puede utilizar entre el cliente y el servidor, no existe una herramienta que pueda realizar estas actividades con todas las transacciones que conforman el escenario y menos aún en los diferentes proyectos de pruebas de desempeño. En estos casos se torna imprescindible desarrollar herramientas o integrar varias de ellas que permitan la automatización de cada transacción. La selección de la o las herramientas adecuadas puede ser de vital importancia en el costo/beneficio del proyecto, ya que ciertas herramientas, del tipo "*world class*" pueden resolver muchos de los problemas más difíciles relacionados con protocolos e incluso hacer muy sencilla la tarea de producción de los scripts, pero su costo puede ser muy alto. En el otro extremo, herramientas "*open source*", de distribución gratuita, pueden demandar mayor esfuerzo de programación e incluso no ser capaces de resolver la comunicación. En caso de que no se encuentre una herramienta adecuada para una determinada transacción, se puede optar por ejecutarla manualmente, aunque hay que tener en cuenta las implicancias que esto conlleva, como son la subjetividad de los tiempos medidos en estas transacciones y la pérdida de la capacidad de reproducir exactamente la misma prueba debido a variaciones en los tiempos humanos.

Más allá de la herramienta seleccionada, es fundamental tener en cuenta que los scripts juegan un papel protagónico en las pruebas de desempeño y un error en los mismos puede implicar sacar conclusiones equivocadas. Por este motivo y debido a que el *testware* es *software*, el proceso propone la realización de revisiones estáticas del código de los scripts (asistidas con listas de chequeo) así como pruebas unitarias de los mismos. También es recomendable seguir normas de código y nomenclatura de archivos para las distintas herramientas.

Al margen de los objetivos de la etapa de automatización, en ocasiones se detectan problemas o posibles mejoras al sistema bajo prueba durante la misma. Si hay algún detalle relevante para reportar, incluso si no es objetivo directo de las pruebas que se están realizando, se reporta con el formato de un *Finding* (hallazgo). Esto es un pequeño memorando que describe el incidente, y es válido para errores funcionales, de concurrencia, seguridad, desempeño, u otro tipo de problemas que quedan visibles al utilizar las herramientas que se usan para generar carga.

2.4 Preparación de la Infraestructura

Esta etapa tiene como objetivo dejar lista toda la infraestructura de pruebas para la ejecución de las mismas, compuesta por todo el ambiente de pruebas, o sea el hardware y software que soportan el sistema bajo prueba y de las generadoras de carga, las conexiones red y los datos de prueba. Si bien se recomienda realizar esta etapa previo a la automatización, debido a que en la etapa de análisis de requerimientos ya se cuenta con la información de la infraestructura, generalmente tiene costos asociados que se minimizan al realizarla justo antes de la ejecución. Estos costos asociados pueden ser licencias de software o disponibilidad del equipo de producción (puede que el mismo se tenga que comprar, alquilar o utilizar para otras actividades).

Dentro del software a instalar, se encuentran las herramientas de monitorización. Monitorizar el comportamiento de la infraestructura y del sistema en general es de primordial importancia en una prueba de desempeño. Estudiar el comportamiento de los servidores, la red e incluso de los propios generadores de carga es de relevancia para saber el por qué del comportamiento de los diferentes componentes de la aplicación. Por ello, además de la generación de la carga, una prueba de desempeño tiene su foco en la monitorización. En esta etapa deben quedar definidas y disponibles las herramientas a ser utilizadas para la monitorización. Cabe recordar que entre los indicadores a tomar durante la prueba, se reconocen los llamados de primer y segundo nivel. Los de primer nivel son generalmente indicadores que se van a monitorizar también cuando el sistema esté en producción, y por tal motivo, son en general de características tales que generan un mínimo esfuerzo extra al sistema y una incidencia mínima en los tiempos de respuesta obtenidos de las transacciones. Estos indicadores que se van a tomar en producción deben estar activos durante la ejecución de las pruebas. En caso que no se prevea monitorizar al sistema en producción, o que la monitorización prevista no brinde la información mínima necesaria para apreciar el desempeño del sistema, las herramientas que se decida usar deberán ser lo mínimo intrusitas posible, para no alterar el modelo de carga. En estos casos, es recomendable medir la incidencia de la herramienta de monitorización durante la etapa de ejecución de los *baseline* [1].

Por otra parte se tienen los indicadores de segundo nivel, los cuales se obtienen de herramientas que generalmente introducen un impacto relevante en el desempeño del sistema y se utilizan para profundizar sobre un problema en particular a la luz de la información arrojada por la monitorización de primer nivel. Como es de suponer, cuando se ejecutan pruebas en las que se tienen activados estos indicadores, los tiempos de respuesta no son tenidos en cuenta, teniendo la prueba su foco en la solución o detección de datos para la corrección de problemas o posibilidades de mejora.

Es importante recordar que las generadoras de carga y la red que conecta dichas máquinas con el sistema bajo prueba, deben de estar también monitorizadas. Si queremos observar el comportamiento bajo condiciones de carga

de una pieza de hardware o software en particular, el eventual cuello de botella debe ser dicho recurso. Esto significa que no se debe introducir otros elementos en la prueba que sean el cuello de botella, como por ejemplo los clientes que generan la carga.

Por otro lado se encuentran los datos de prueba. Cada repetición de la prueba a ejecutar debe hacerse con el mismo juego de datos y para esto se debe establecer entonces un estado inicial de los datos, y tener la capacidad de volver al mismo antes de cada iteración, contando con un procedimiento bien definido para realizar las tareas de respaldo y restauración de los mismos en el caso de que los datos se consuman en la ejecución de las pruebas.

2.5 Ejecución y Reportes

Luego de que se han completado las etapas anteriores, se tienen todos los artefactos necesarios para ejecutar las pruebas de desempeño. Si bien al final de esta etapa se deben haber ejecutado los escenarios previstos en la etapa de análisis de requerimientos, la metodología propone realizar varias ejecuciones en las cuales la carga sobre el sistema se va incrementando de manera gradual hasta llegar a los escenarios objetivos o superarlos.

Previo a la ejecución de los escenarios, se deben ejecutar los denominados *baselines* (línea de base). La ejecución de los baselines consiste en ejercitar cada transacción seleccionada con un único usuario virtual y con toda la infraestructura dedicada al mismo. El objetivo es conocer los mejores tiempos que se llegar a obtener en cada transacción y de esta manera tener un punto de referencia para analizar el impacto de los distintos escenarios de concurrencia sobre cada transacción. Debido a que estos tiempos son de fundamental importancia para futuros análisis, es necesario ejecutar cada transacción una cantidad suficiente de veces como para obtener datos estadísticamente válidos.

Como se mencionaba anteriormente, la ejecución de los escenarios definidos se realiza de manera gradual. Se debe definir como se incrementará la carga de acuerdo a la realidad del proyecto, aunque generalmente se aplica un incremento del 20% en cada prueba. Por este motivo se ejecuta el 20% de la carga, luego el 40%, el 60%, el 80% hasta llegar al 100% que es el escenario objetivo. Esta estrategia de ejecución que propone el proceso trae aparejado varias ventajas. La primera es que los problemas de desempeño más evidentes del sistema aparecen en porcentajes de carga bajos en donde la complejidad de la prueba es menor. Luego a medida que se incrementa la carga se van atacando problemas cada vez más sutiles hasta llegar a cumplir los objetivos planteados. Por otro lado, en las primeras ejecuciones, de baja carga, se van ajustando algunos detalles de infraestructura tales como las herramientas de monitorización, los datos de prueba o incluso los propios scripts de prueba. También el personal involucrado en las pruebas va adquiriendo la metodología de ejecución de manera paulatina, lo cual es preferible antes que comenzar con la complejidad que puede implicar el escenario objetivo.

La *Fig. 2* muestra la dinámica que se sigue en esta etapa para cada una de las ejecuciones.

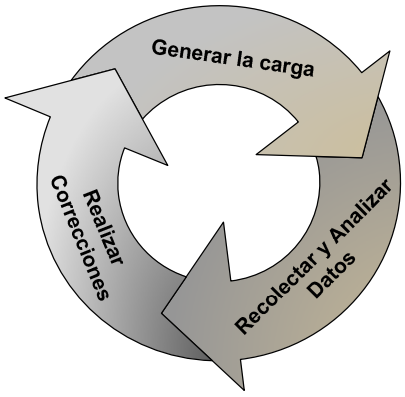

Fig. 2. Ciclo de ejecuciones

Luego de que se ejecutó cada escenario con todos los monitores definidos activados, se debe recolectar y analizar la información. Es importante prestar atención a los distintos datos, tanto del lado de las generadoras de carga como del resto de la infraestructura. El dato más insignificante puede ser la causa de un problema y en esta materia habitualmente se cumple el principio del 80/20 en donde el 20% de los datos nos proporciona el 80% de la información. Por este motivo es de vital importancia para el éxito del proyecto contar con expertos en cada uno de los componentes con el fin de poder rápidamente focalizarse en ese 20% de la información relevante. Es necesario en este tipo de pruebas poder interrelacionar los distintos indicadores para analizar globalmente el sistema y localizar

cual es la pieza que está convirtiéndose en el cuello de botella. Hay que tener en cuenta que los tiempos de respuesta son sólo los síntomas, el desafío es encontrar las causas de esos síntomas.

Más allá de los indicadores recopilados, el proceso sugiere que un experto funcional de la aplicación utilice el sistema durante la prueba. Esto tiene como objetivo obtener una retroalimentación subjetiva del comportamiento del sistema, ya que muchas veces cuesta visualizar si el tiempo en segundos es adecuado o no para la operativa del negocio. A esto se le da en el proceso el nombre de *monitor humano*.

Luego que se ha analizado toda la información resultante de las pruebas y confirmado los datos obtenidos, se debe comunicar al resto de las partes involucradas. Con este fin se puede preparar un reporte formal con el detalle de toda la información y/o pequeños informes o gráficas que varíen de acuerdo al público que recibirá dicho material. Hay que recordar que el personal involucrado en una prueba de desempeño tiene conocimientos heterogéneos y espera recibir información útil de la prueba de acuerdo a sus intereses.

Finalmente, se realizan correcciones y se ajusta el sistema para una próxima corrida de pruebas. Las mejoras al desempeño del sistema se pueden encarar principalmente por dos vías: mejoras en la lógica o ajustes a la infraestructura. El análisis de los datos tiene que haber arrojado información suficiente para saber por dónde abordar la corrección o mejora.

2.6 Etapa Final

En esta etapa se le da un cierre al proyecto de pruebas de desempeño tanto para el equipo de pruebas como para el cliente que contrató el proyecto.

Con este fin se realiza un informe final del proyecto en el cual se detalla la metodología seguida a lo largo del mismo, las ejecuciones realizadas, las mejoras efectuadas sobre el sistema y una serie de recomendaciones para seguir mejorando el desempeño del sistema. Otro punto importante de esta etapa es realizar una recopilación de toda la información y artefactos de prueba creados a lo largo del proyecto con el fin de entregárselos al cliente. Junto con la entrega de este material, es recomendable realizar una presentación para una transferencia tecnológica de las herramientas utilizadas y de la metodología seguida. Esta actividad tiene el objetivo de proveer al cliente el conocimiento necesario para que pueda seguir con la ejecución de las pruebas en caso de así requerirlo.

Posteriormente, el equipo de pruebas realiza el denominado análisis post-mortem. El mismo consiste en una evaluación general del proyecto para analizar las desviaciones con respecto a la estimación, el cumplimiento de los objetivos del proyecto y las oportunidades de mejora de la metodología seguida.

3 Caso de Estudio - Sistema Bancario

En esta sección se analizará la aplicación de la metodología propuesta en un caso concreto realizado recientemente por el Centro de Ensayos de Software.

3.1 Contexto general del proyecto

El caso de estudio estuvo enmarcado dentro de un proceso de migración del sistema central de una entidad bancaria a un producto nuevo, con nueva arquitectura y nuevas funcionalidades. La migración tuvo una duración de más de un año e involucró a un centenar de personas. En este contexto se exigía la realización de pruebas de desempeño antes de la puesta en producción debido a que fallas en los tiempos de respuesta o en la estabilidad del sistema podrían ocasionar cuantiosas pérdidas a la institución.

Una particularidad del proyecto es que parte del trabajo se realizó a distancia. El CES tiene sus laboratorios en Montevideo, Uruguay, y el cliente reside en Buenos Aires, Argentina. Esto implicó para todas las etapas viajes periódicos, pero la mayor parte del trabajo se realizó a distancia. El equipo de prueba estaba compuesto por un líder de proyecto, cuatro *testers seniors* y un *tester junior*.

3.2 Etapa Inicial

Siguiendo la metodología, inicialmente se evaluó el esfuerzo del proyecto de acuerdo a cada actividad propuesta en el proceso. Este enfoque permitió centrarse en los principales parámetros que determinan una prueba de desempeño para estimar de manera sencilla y ajustada el esfuerzo de cada rol del equipo de pruebas. Para algunas tareas en particular (como por ejemplo la automatización de transacciones) es de vital importancia realizar pruebas de concepto que permitan mitigar riesgos técnicos y evaluar la dificultad de realizar dicha tarea. En este caso en particular el equipo de testing ya había realizado pruebas sobre la misma aplicación por lo que conocía el esfuerzo requerido para su automatización. La planificación para el mismo se muestra en la *Fig. 3*.

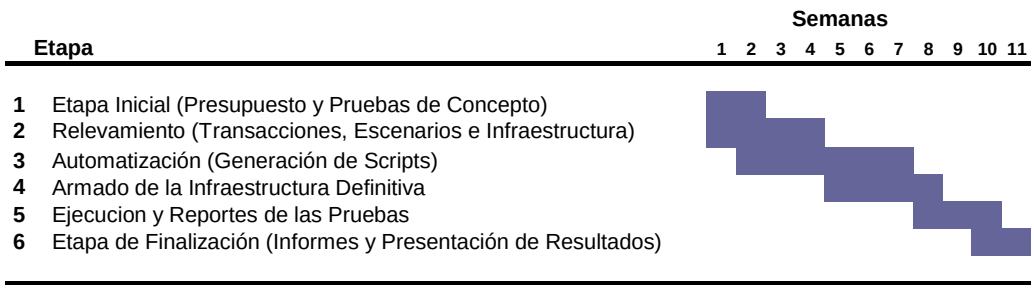

Fig. 3. Cronograma del proyecto

3.3 Análisis de Requerimientos

En esta etapa se procedió a analizar con profundidad las transacciones que se incluirían en las pruebas, los escenarios, datos e infraestructura. Para realizar esta tarea, tuvieron lugar varías reuniones con los responsables funcionales de la entidad bancaria y con los técnicos que desarrollaron y personalizaron la aplicación. Con todos los involucrados se analizaron cada uno de los puntos basándose en los lineamientos que sugiere la metodología. Se seleccionaron un conjunto de catorce transacciones accedidas por interfaz Web, tres transacciones accedidas por Web Services [4] y dos accedidas por ISO8583 [5]. Dichas transacciones corresponden al escenario objetivo definido, el cual simula la realidad de la entidad financiera en la hora pico de su operatoria.

3.4 Automatización

En esta etapa se procedió a la creación de los scripts que automatizan las transacciones. Si bien la metodología sugiere que la versión de la aplicación este congelada, esto no se pudo lograr debido a la proximidad entre la fecha de salida en producción y el fin del desarrollo del sistema. Esto trajo aparejado retrasos en la etapa de automatización debido a la necesidad de adaptar los scripts generados a nuevas versiones de la aplicación que se iban liberando.

Para poder realizar la automatización se utilizaron varias herramientas de generación de carga. El sistema bajo prueba conecta los clientes a través de tres protocolos: HTTP, HTTP/SOAP [4] e ISO-8583 [5]. Los protocolos Web fueron automatizados con la herramienta OpenSTA [6] mientras que para la simulación de los ATM a través de ISO-8583 se utilizó una herramienta desarrollada por el Centro de Ensayos de Software denominada ISOLoadGenerator.

Una de las transacciones definida inicialmente no se pudo automatizar debido a que su finalización se produjo de manera muy próxima a la ejecución de las pruebas. Por tal motivo se decidió ejecutar dicha transacción de manera manual por un usuario idóneo.

3.5 Infraestructura

Al preparar el ambiente definitivo para las pruebas se decidió incluir las nuevas versiones del sistema y se realizó una migración de la base de datos. Como sucede generalmente, esto derivó en la necesidad de ajustar los scripts.

El sistema a implantar es un sistema *Full Web*, y fue construido con una herramienta de generación de código. En este caso el sistema fue generando en *Java*, y corre sobre un cluster de cinco servidores de aplicación *WebSphere* y utiliza una base de datos *DB2*en un *iSeries 570*. Los servidores de aplicación se conectaban a través de un balanceador de carga *CISCO* CSS11500. Por otro lado, el sistema principal se conectaba con sistemas externos, alguno de los cuales también formaron parte del sistema bajo pruebas definido.

3.6 Ejecución

Los ejecuciones planteadas según la metodología eran una inicial para los Baselines y luego ir escalando la carga desde el 20%, 40%, 60%, 80%,100% hasta llegar al 120%.

Si bien la metodología implica no pasar de un "escalón" a otro hasta que se compruebe el correcto funcionamiento del sistema en cada escalón, debido a las necesidades y tiempos del proyecto se tornó necesario saltar algunos de los escalones. Por este motivo finalmente se ejecutaron los Baselines, luego el 20%, luego se ejecutó el 50% y el 75% sin lograr ejecuciones completamente satisfactorias y por último se ejecutaron satisfactoriamente el 100% y el 150%.

Tal como describe la metodología, este enfoque permitió encontrar problemas graves (u oportunidades de mejora importantes) de manera temprana y con una complejidad menor en los escenarios de prueba, lo que logró acelerar la corrección de los mismos. Para cada ejecución relevante se entregó un informe con los principales indicadores y evaluaciones de cada ejecución. También se utilizó una bitácora para registrar cada ejecución y cada cambio realizado al sistema. Esta herramienta es imprescindible a la hora de analizar toda la etapa de ejecuciones.

3.7 Etapa final

Como punto principal de la etapa final, se elaboró y entregó al cliente el informe final, el cual cuenta todos los detalles relevantes del proyecto. También se entregó el resto del material generado y se culminó con una presentación ante todos los involucrados haciendo un resumen del proyecto de testing

En la instancia de postmortem se analizó el proyecto en su globalidad, analizando las estimaciones iniciales contra el desenlace que finalmente tuvo el proyecto. Este punto se detalla en la próxima sección (*Planificado vs. Realizado*).

Planificado vs. Realizado. Si bien ocurrieron algunas modificaciones en el alcance inicial de la prueba, el proyecto se desarrollo con un esfuerzo estimado (1.010 horas persona) muy próximo al realizado (1.080 horas persona) y con un defasaje en el calendario de cuatro semanas. Este estrecho margen de diferencia en el esfuerzo planificado fue posible gracias al proceso seguido y a la experiencia en la utilización de dicho proceso. Por otro lado el defasaje en el calendario se debió principalmente a tiempos muertos a raíz de no poder contar con la infraestructura (por ejemplo en la semana 14 no se pudo realizar ninguna tarea) y por retrasos en la finalización del desarrollo y pruebas funcionales de la aplicación, lo cual tuvo impacto en re-trabajos de automatización de las transacciones.

En la *Fig. 4* se muestra un esquema del tiempo de la ejecución real del proyecto.

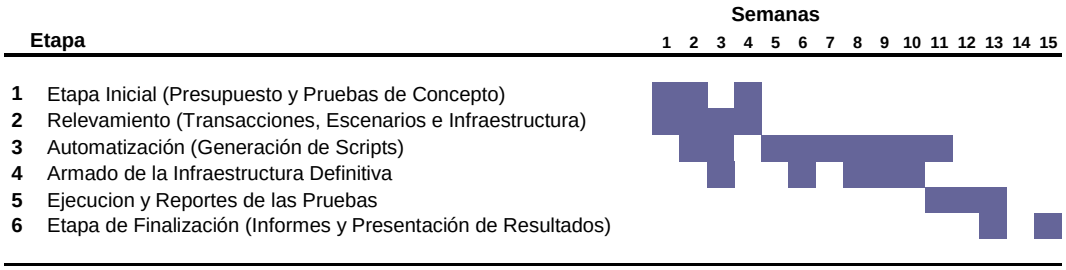

Fig. 4. Cronograma real del proyecto

Si se compara con la *Fig. 3* se puede observar que las etapas si bien insumieron tiempos cercanos a los estimados, se realizaron de forma más discontinuada que en la planificación. Esta característica es muy común en los proyectos de desempeño debido a que se requiere de una contraparte muy fuerte que pueda brindar respuestas a los problemas y desafíos que vayan surgiendo.

4 Conclusiones

Este proceso se ha aplicado desde los orígenes del Centro de Ensayos de Software en el año 2005, y se ha ido mejorando desde entonces. Se ha utilizado en más de 10 proyectos sobre sistemas que atacan diferentes realidades de negocio, y la metodología es aplicable en forma efectiva para el caso de una empresa que realiza testing de desempeño en forma tercerizada o también para aquellas empresas productoras o consumidoras de software. Si bien la metodología define una base para las tareas a desarrollarse, es imprescindible contar con un personal altamente capacitado y con amplitud de conocimiento así como también con expertos técnicos que conozcan al detalle cada parte del sistema bajo prueba. Esto permitirá no sólo que el proyecto cumpla con las expectativas de costo y beneficio, sino también que esta metodología pueda ser adaptada y utilizada en diferentes procesos de desarrollo.

El proceso sugerido permite tempranamente estimar de manera ajustada y práctica el esfuerzo del proyecto de pruebas de desempeño, y luego a medida que transcurre el proyecto, permite en cada momento centrarse en los puntos calves para realizar un proyecto exitoso.

También permite saber de manera clara cuales son las condiciones que debe cumplir el cliente en cada etapa de las pruebas de desempeño, lo que permite organizar un cronograma con las obligaciones de cada parte.

Esta metodología cumple con los objetivos que se plantearon al momento de la concepción del proyecto de caso de estudio. En el caso de estudio presentado, los principales beneficios obtenidos para la entidad financiera gracias a que la metodología permitió que se pueda realizar un proyecto exitoso de pruebas de desempeño fueron los siguientes:

- Minimización de riesgos en cuanto a desempeño del sistema al salir en producción.
- Anticipación a una serie importante de problemas que iban a suceder en producción dando la oportunidad de encontrar mejoras con mayor calma y menos costos
- Capacitación de los técnicos de la propia entidad.

- Agilización de medidas a tomar ante casos de emergencia.
- Dimensionamiento de hardware necesario para la carga actual del sistema (al IBM iSeries se le adicionaron 4500CPW durante el proyecto, obteniéndose un beneficio claro en el incremento de la capacidad de procesamiento de esta infraestructura)

5 Trabajos Futuros

Dentro de la mejora continua de la metodología se plantea la definición de métricas que den soporte a la toma de decisiones gerenciales del proyecto de pruebas de desempeño. Por ejemplo, la incorporación del concepto de valor ganado [2] para el análisis de situación en las etapas críticas del proceso, como ser automatización y ejecución. Esto dará mayor sustento al momento de negociar con el cliente temas de alcance del proyecto, ya que proporcionará números aproximados que permiten cuantificar en forma más tangible la situación en base a costos, estimación y ejecución.

Asimismo, se busca la incorporación de conceptos que permitan estimar, de alguna manera, los tiempos muertos que se tendrán en la corrección y mejoras de los problemas encontrados. Este punto ha sido el mayor problema a la hora de comparar los tiempos finales del proyecto con los tiempos estimados en un inicio, si bien, como en el ejemplo planteado en el caso de estudio, el esfuerzo realizado por el Centro de Ensayos de Software se acerca en ambas estimaciones.

Por último, el concepto de cuando parar de realizar pruebas es un punto que puede llegar a influir en gran manera en el tema de la estimación de recursos y tiempos. Si bien en las etapas tempranas del proceso se definen ciertas métricas orientadas a definir este punto, una vez que se las pruebas se encuentran en ejecución es necesario contraponer estas métricas con las obtenidas en la ejecución del proyecto y en las expectativas del cliente del servicio.

Referencias

1. J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, Dennis Rea: Performance Testing Guidance for Web Applications. Microsoft Corporation, September 2007.
2. Project Management Institute (PMI). Guía de los Fundamentos de la Dirección de Proyectos (PMBOK) (3ª ed.). PMI 2004.
3. Centro de Ensayos de Software (CES), www.ces.com.uy
4. W3C, Web Services Activity, http://www.w3.org/2002/ws/
5. International Organization for Standardization (ISO), www.iso.org.
6. OpenSTA, www.opensta.org.
7. Beatriz Pérez. “Proceso de Testing Funcional Independiente (ProTest)”. Tesis de Maestría en Informática, PEDECIBA Informática, Instituto de computación, Facultad de Ingeniería, Universidad de la Republica, Uruguay, ISSN: 0797-6410 - 06-11, 2006.
8. Guide to the Software Engineering Body of Knowledge - SWEBOK, 2005 version. IEEE Computer Society. www.swebok.org
9. Kit E., Software Testing In The Real World: Improving The Process, Addison Wesley. ISBN 0201877562
10. Scott Barber: User experience, not metrics. www.perftestplus.com
11. Steven Haines: Performance Testing Methodology. Quest Software. Octubre 2005.