Automatización y Gestión de las Pruebas Funcionales usando Herramientas Open Source

Ignacio Esmite, Mauricio Farías, Nicolás Farías, Beatriz Pérez
Centro de Ensayos de Software (CES), Universidad de la República
Montevideo, Uruguay, 11000
{iesmite, mfarias, nfarias, bperez}@fing.edu.uy

Resumen

En este artículo se presenta una metodología y el conjunto de herramientas open source utilizado para la automatización de las pruebas funcionales de productos con interfaz web. Este conjunto de herramientas está compuesto por: Selenium, Eclipse y extensiones de Mozilla Firefox como son Firebug, XPath Checker y XPather. Se describe la experiencia de utilizar la metodología en un proyecto de automatización específico y se concluye la factibilidad para la automatización de las pruebas siguiendo las actividades y el conjunto de herramientas definidos. Si bien las herramientas asisten en las pruebas automatizadas, no brindan soporte para la organización de los artefactos del proyecto: scripts, documentos y reportes de ejecución. Como trabajo a futuro se propone integrar al conjunto de herramientas definido, la herramienta FitNesse para gestionar los artefactos, buscando mejorar la organización de las pruebas junto con la comunicación y colaboración del equipo de pruebas.

Palabras claves: Ingeniería de software, Pruebas, Pruebas Funcionales, Pruebas Funcionales Automatizadas, Herramientas, Open Source**.**

1 INTRODUCCIÓN

La automatización de las pruebas funcionales reduce significativamente el esfuerzo dedicado a las pruebas de regresión en productos que se encuentran en continuo mantenimiento. La automatización de las pruebas debe ser considerada un proyecto en sí mismo con objetivos definidos.
En este artículo se presenta una metodología y el conjunto de herramientas open source utilizado para la automatización de las pruebas funcionales en productos con interfaz Web. Este conjunto de herramientas está compuesto por: Selenium[22], Eclipse[8] y extensiones de Mozilla Firefox[17] como son Firebug[9], XPath Checker[29] y XPather[30]. Se describe la experiencia de utilizar la metodología en un proyecto de prueba específico y se concluye la factibilidad para la automatización de las pruebas siguiendo las actividades y el conjunto de herramientas definidos. Si bien las herramientas asisten en las pruebas automatizadas, no brindan soporte para la organización de los artefactos del proyecto: scripts, documentos y reportes de ejecución. Como trabajo a futuro se propone integrar al conjunto de herramientas definido, la herramienta FitNesse[10] para gestionar los artefactos, buscando mejorar la organización de las pruebas junto con la comunicación y colaboración del equipo de pruebas.
En la sección 1.1 se describen los principales conceptos relacionados con las pruebas, en la sección 1.2 se introducen los conceptos de herramientas de automatización. Luego en la sección 1.3 se describe el contexto y motivación para este trabajo.
En la sección 2 se describe la experiencia práctica de automatizar las pruebas usando Selenium y otras herramientas open source.
En la sección 3 se muestra una propuesta para gestionar los productos resultantes de la automatización utilizando la herramienta FitNesse. Por último, en la sección 4 se presentan las conclusiones del artículo.

1.1 Principales Conceptos

Se presentan los principales conceptos relacionados con la automatización de las pruebas funcionales. Se define prueba funcional, caso de prueba, procedimiento de prueba, script de prueba, suite de prueba, pruebas de regresión y pruebas de humo.

El objetivo de la prueba funcional es validar cuando el comportamiento observado del software probado cumple o no con sus especificaciones. La prueba funcional toma el punto de vista del usuario [2]. Las funciones son probadas ingresando las entradas y examinando las salidas. La estructura interna del programa raramente es considerada [16].
Para realizar pruebas funcionales, la especificación se analiza para derivar los casos de prueba. Técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto y conjetura de errores son especialmente pertinentes para las pruebas funcionales. Se deben considerar condiciones inválidas e inesperadas de la entrada y tener en cuenta que la definición del resultado esperado es una parte vital de un caso de prueba. El propósito de la prueba funcional es mostrar discrepancias con la especificación y no demostrar que el programa cumple su especificación [18].

Un caso de prueba (test case) es un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y poscondiciones de ejecución, desarrollados con un objetivo particular o condición de prueba, tal como ejercitar un camino de un programa particular o para verificar que se cumple un requerimiento específico [12].

Un script de prueba (test script) son los datos y las instrucciones escritas con una sintaxis formal, almacenado en un archivo y usado por una herramienta de automatización de las pruebas. Un script de prueba puede automatizar uno o más casos de prueba, navegación, inicialización u operaciones de configuración del entorno. Un script de prueba previsto para la ejecución manual de las pruebas es un procedimiento de prueba [26].
Una suite de pruebas es uno o más conjuntos de pruebas reunidos para satisfacer un objetivo de prueba [26]. Un conjunto de prueba incluye scripts y documentación. En nuestro caso, las suites constituyen un conjunto de scripts y el orden de ejecución de los mismos.

Las pruebas de regresión tienen como objetivo verificar que no ocurrió una regresión en la calidad del producto luego de un cambio, asegurándose que los cambios no introducen un comportamiento no deseado o errores adicionales. Implican la reejecución de alguna o todas las pruebas realizadas anteriormente [3].
Las pruebas de humo son un conjunto de pruebas aplicadas a cada nueva versión, su objetivo es validar que las funcionalidades básicas de la versión se comportan según lo especificado. Estas pruebas buscan grandes inestabilidades o elementos clave faltantes o defectuosos, que hacen imposible realizar las pruebas como fueron planificadas para la versión. Si la versión no pasa las pruebas de humo, no se comienza la ejecución de las pruebas planificadas de la versión [15].

1.2 Automatización de las pruebas

Hay herramientas que apoyan diversos aspectos de la prueba. A continuación se presenta una clasificación posible para las herramientas [13]:

- Administración de las pruebas y el proceso de pruebas: herramientas para la administración de las pruebas, para el seguimiento de incidentes, para la gestión de la configuración y para la administración de requerimientos.
- Pruebas estáticas: herramientas para apoyar el proceso de revisión, herramientas para el análisis estático y herramientas de modelado.
- Especificación de las pruebas: herramientas para el diseño de las pruebas y para la preparación de datos de prueba.
- Ejecución de las pruebas: herramientas de ejecución de casos de prueba, herramientas de pruebas unitarias, comparadores, herramientas de medición del cubrimiento, herramientas de seguridad.
- Desempeño y monitorización: herramientas de análisis dinámico, herramientas de desempeño, de carga y de estrés, herramientas de monitorización

Para la automatización de las pruebas funcionales son especialmente indicadas las herramientas de ejecución de las pruebas de captura y reproducción. Estas herramientas permiten al tester capturar y grabar pruebas, para luego editarlas, modificarlas y reproducirlas en distintos entornos. Herramientas que graban la interfaz de usuario a nivel de componentes y no de bitmaps son más útiles. Durante la grabación se capturan las acciones realizadas por el tester, creando automáticamente un script en algún lenguaje de alto nivel. Luego el tester modifica el script para crear una prueba reusable y mantenible. Este script se vuelve la línea base y luego es reproducido en una nueva versión, contra la cual es comparado. En general estas herramientas vienen acompañadas de un comparador, que compara automáticamente la salida en el momento de ejecutar el script con la salida grabada [7].

1.3 Contexto y Motivación

El Centro de Ensayos de Software (CES) [5] es un emprendimiento conjunto de la Universidad de la República de Uruguay (UdelaR) [28] y de la Cámara Uruguaya de Tecnologías de la Información (CUTI)[4], entidad que agrupa a la mayoría de las empresas productoras de software del país. Los servicios que ofrece el CES incluyen

- Servicios de prueba independiente: Planificar, diseñar, coordinar y ejecutar pruebas de productos de software de manera efectiva y controlada, definiendo claramente el contexto y los objetivos.
- Consultoría: Asesorar a las organizaciones en la mejora de los procesos de prueba, definición de estrategias y automatización de las pruebas. Colaborar en la creación y consolidación de sus áreas de prueba.
- Capacitación: Elaborar e impartir programas de capacitación en la disciplina de testing según las necesidades de cada organización.

Alguno de los clientes del CES, son empresas uruguayas medianas y pequeñas productoras de tecnologías de la información que muestran gran interés en automatizar sus pruebas funcionales. Gran parte de estas empresas comercializan uno o dos productos de software, los cuales personalizan para su venta en distintos clientes, debido a esto, los productos están en continuo mantenimiento y mejora. Con cada nueva versión del producto, las empresas necesitan asegurarse que las principales funcionalidades del producto continúan operando correctamente. Las empresas ven la importancia de utilizar la automatización para reducir los costos y tiempos en las pruebas de regresión y poder contar con un conjunto de pruebas de humo automatizadas. Muchas de ellas, buscan que el costo de la licencia no sea una barrera para comenzar el proceso de automatización. Esta es una de las razones que hacen más atractiva para ellas la automatización con herramientas open source.
Al momento de seleccionar una herramienta para las pruebas, las empresas buscan aquella que con el menor costo, les permita construir las pruebas automatizadas que mejor se adapten a su producto. La confianza en la herramienta cumple un papel fundamental. Para esto no solamente requieren tener personal con experticia en la herramienta, también tienen que asegurarse de que la misma podrá ser extendida para resolver nuevos problemas en caso de ser necesario. Para ello es muy importante evaluar el uso, la información y el soporte que tiene la herramienta en la comunidad.

2 AUTOMATIZACIÓN DE LAS PRUEBAS FUNCIONALES

Existe un proceso definido para las pruebas funcionales manuales en el Centro de Ensayos de Software. Dicho proceso cuenta con etapas, actividades, roles y artefactos definidos para un proyecto de prueba independiente, donde el equipo de pruebas es contratado para realizar las pruebas de un producto de software desarrollado por terceros. El proceso utilizado se llama ProTest y puede ser consultado en [21].
La experiencia en proyectos de automatización de pruebas funcionales motivó la necesidad de extender el proceso definido, con nuevas actividades específicas para la automatización.
En la sección 2.1 se presentan las actividades específicas definidas para la automatización de las pruebas. En la sección 2.2 se describen las herramientas open source utilizadas para dicha automatización, en la sección 2.3 se describe la experiencia en el CES de realizar las actividades con estas herramientas. Por último en la sección 2.4 se presentan las conclusiones sobre la utilización de herramientas open source para la automatización de las pruebas funcionales de productos de software.

2.1 Metodología Propuesta

A continuación se describen las actividades que utiliza el CES en sus proyectos de automatización. Se especifican los objetivos y se detallan las tareas realizadas en cada actividad. En la Figura 1 se muestra el diagrama de actividad para las actividades definidas y en la Tabla 1 se resumen los roles involucrados y los artefactos de entrada y de salida de cada actividad.

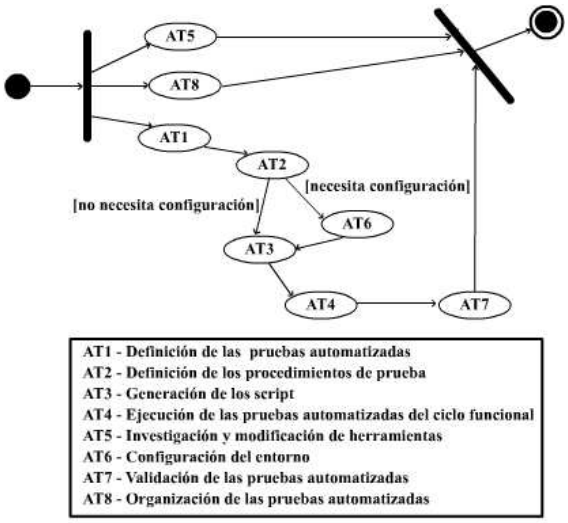

Figura 1 – Diagrama de Actividad

AT1 – Definición de las pruebas automatizadas
El objetivo de esta actividad es definir el conjunto de ciclos funcionales o funcionalidades que se probarán con las pruebas automatizadas.
Esta actividad consiste en definir con el Cliente qué pruebas se van a automatizar (en alto nivel), considerando los ciclos funcionales o funcionalidades que involucran. Para ello se priorizan y evalúan las pruebas y las funcionalidades a probar.

AT2 – Definición de los Procedimientos de Prueba
El objetivo de esta actividad es definir las suites y scripts que conformarán las pruebas automatizadas.
A partir de los ciclos funcionales y funcionalidades seleccionados para las pruebas, se especifican las suites y scripts que las ejecutarán. Definir las suites implica definir los scripts que las componen y especificar posibles dependencias de ejecución entre ellas. Para cada script se debe definir la tarea que debe realizar y las verificaciones que debe contener.

AT3 – Generación de suites y scripts
El objetivo de esta actividad es obtener las suites con sus scripts correspondientes.
Esta actividad consiste en el armado de cada una de las suites correspondientes al ciclo funcional a automatizar. Se graba o codifica las pruebas obteniendo como resultado los scripts integrantes de las suites. Se verifica el correcto funcionamiento de cada script.

AT4 – Ejecución de las pruebas automatizadas del ciclo funcional
El objetivo de esta actividad es ejecutar una prueba completa de las suites correspondientes al ciclo funcional y verificar su correcto funcionamiento.

Esta actividad consiste en realizar la prueba completa del ciclo funcional en el entorno preparado para dicho fin. En caso de un funcionamiento incorrecto deben realizarse los ajustes necesarios. En esta actividad se verifica el comportamiento de las suites en su conjunto.

AT5 – Investigación y modificación de herramientas

El objetivo de esta actividad es encontrar soluciones a las necesidades que no pueden satisfacerse con las herramientas de automatización que se manejan.

Esta actividad incluye recorrer foros, referencias y buscar antecedentes similares a la necesidad planteada. Luego se analizan las posibles soluciones. Estas incluyen instalar nuevas versiones de las herramientas, modificar las herramientas o extenderlas. Esta actividad también incluye búsqueda de nuevas herramientas que asistan en la automatización.

AT6 – Configuración del entorno

El objetivo de esta actividad es configurar el entorno de la aplicación que se desea probar para poder ejecutar las suites correctamente y documentar esta configuración.

En esta actividad se debe configurar los datos que mantiene la aplicación de manera que permitan la correcta ejecución de las suites. Es importante documentar con el detalle suficiente esta configuración.

ACTIVIDAD	ROLES	ENTRADA	SALIDA
AT1 - Definición de las pruebas automatizadas	Líder Diseñador de Pruebas Cliente	Requerimientos Acta reunión cliente	Documento que especifica las funcionalidades y ciclos funcionales a probar con las pruebas automatizadas
AT2 - Definición de los procedimientos de prueba	Diseñador de Pruebas Cliente	Requerimientos Acta reunión cliente Documento obtenido en AT1	Documento con Suites y Scripts que componen las pruebas
AT3 - Generación de suites y scripts	Tester	Documento obtenido en AT2 y documento obtenido en AT6	Suites y Scripts que las componen
AT4 - Ejecución de las pruebas automatizadas del ciclo funcional	Tester	Suites y Scripts obtenidos en AT3	Suites y Scripts verificados
AT5 - Investigación y modificación de herramientas	Tester	Herramienta a investigar	Nuevas herramientas (adquiridas o modificadas) y documentación pertinente
AT6 - Configuración del entorno	Tester	Aplicaciones a probar y documento obtenido en AT2	Aplicaciones aptas para ser probadas y documentación pertinente
AT7 - Validación de las pruebas automatizadas	Cliente Tester	Suites y Scripts obtenidos en AT4	Suites y Scripts verificados y validados
AT8 - Organización de las pruebas automatizadas	Diseñador de Pruebas Tester	Artefactos generados en el proyecto	Artefactos gestionados

Tabla 1 – Actividades, Roles y Artefactos de Entrada y de Salida

AT7- Validación de las pruebas automatizadas

El objetivo de esta actividad es verificar el correcto comportamiento de los scripts, en el ambiente de prueba del cliente y preparar las suites y scripts generados para la validación del cliente.

En esta actividad se prueba y eventualmente se ajustan los scripts para el correcto funcionamiento en el ambiente de pruebas del cliente. El cliente valida comparando las pruebas que espera automatizar con las pruebas que los scripts realizan.

AT8 – Organización de las pruebas automatizadas

El objetivo de esta actividad es gestionar los artefactos generados en el proyecto de automatización.

Esta actividad consiste en definir, actualizar y ejecutar los procedimientos para la gestión de los documentos, suites y scripts que se generen en el proyecto de automatización.

2.2 Herramientas utilizadas

El grupo de herramientas Selenium que ayudan en la ejecución de pruebas, está constituido por Selenium Core[23], Selenium IDE[24] y Selenium Remote Control[25]. Estas herramientas permiten crear y ejecutar pruebas automatizadas sobre aplicaciones Web. Las pruebas automatizadas pueden ser utilizadas como pruebas de regresión o para probar la aplicación sobre diferentes plataformas y navegadores Web. Cada una de estas herramientas son el producto de proyectos que continúan en desarrollo. Estos son llevados a cabo por la Comunidad Open QA[20] y utilizan la licencia Apache 2.0 [1]

La herramienta principal de Selenium es Selenium Core. Es una herramienta de automatización desarrollada inicialmente por programadores y testers de ThoughtWorks [27] y actualmente por OpenQA. Es una aplicación Web desarrollada con HTML y Javascript que soporta una gran variedad de plataformas y navegadores. Permite ejecutar una prueba o conjunto de pruebas automatizadas.

Selenium IDE es una herramienta de grabación que permite grabar pruebas para ejecutarlas en Selenium Core y se integra como una extensión en Mozilla Firefox . Registra las acciones que ejecuta el usuario a través del navegador, generando un script de prueba. Luego permite editarlo.

Una alternativa a Selenium Core es Selenium Remote Control. Esta herramienta de automatización brinda interfaces de programación (APIs) para lenguajes como Java, .NET, Perl, Python y Ruby. A través de ellas los programas pueden interactuar directamente con la interfaz Web de la aplicación utilizando navegadores como Mozilla Firefox e Internet Explorer[14].

Como entorno de desarrollo de los scripts se utilizó Eclipse, que brinda facilidad para crear y editar los scripts utilizando *plug-ins* para edición de HTML y XML. Además permite manejar el control de versiones de forma integrada, ofreciendo un cliente para conectar con el repositorio de scripts (CVS[6]). Cabe destacar que se hace uso de un conjunto mínimo de las funcionalidades que este entorno de desarrollo provee.

El navegador Web Mozilla Firefox permite integrar varias aplicaciones que ayudan a la creación de scripts. Entre ellas destacamos Firebug, XPath Checker y XPather.

La aplicación Firebug pone a disposición herramientas que permiten visualizar y editar fácilmente CSS, HTML y Javascript de la página en la cual se navega. También se puede ejecutar paso a paso el código Javascript de la página.

Por otro lado, XPath Checker y XPather identifican y muestran elementos de la página a través de su XPath. También pueden utilizarse de modo inverso, haciendo clic derecho sobre los elementos de la pantalla para poder obtener su XPath.

2.3 Experiencia

Desde que el CES inició la investigación en el área de automatización de pruebas funcionales, hace un año atrás, se han realizado varios proyectos donde se han automatizado pruebas funcionales para aplicaciones Web. Una de estas aplicaciones, desarrollada con Genexus [11], fue probada tanto en su versión para la plataforma Java como para la plataforma .NET. Otra de las aplicaciones para las cuales se automatizaron pruebas fue construida utilizando Java. En general, con mayor y menor dificultad se pudo lograr la automatización.

En el proyecto de automatización de las pruebas para la aplicación desarrollada con Genexus, el equipo de automatización que llevo a cabo este proyecto estuvo conformado por tres testers automatizadores, que cumplían también el rol de diseñador de pruebas y un líder de proyecto. El cliente quería automatizar un subconjunto de pruebas de regresión, para ejercitar los caminos y ciclos funcionales típicos de la aplicación. Se definieron dos etapas para la automatización, donde se automatizaron distintos escenarios de prueba. En la etapa 2 se extendieron las pruebas de la etapa 1 y se crearon pruebas para nuevos escenarios. Las dos etapas duraron dos y tres meses

respectivamente, en cada una de estas etapas se trabajó aplicando la metodología propuesta. En la tabla 2 se puede observar la cantidad de suites y scripts generados en cada etapa. Si apreciamos la relación entre la cantidad de suites y scripts generados y el tiempo de cada etapa, observamos que a medida que se fue adquiriendo experiencia en el uso de Selenium, la productividad del personal fue creciendo, lo que marca la necesidad de tener testers automatizadores especializados en el área. Un punto importante a destacar en este proyecto en particular, es el alto porcentaje de reutilización de los scripts de una etapa a la otra. De los 271 scripts generados en la segunda etapa, el 40% se basó en scripts generados en la primera etapa, esto contribuyó también al aumento de productividad. Los scripts reutilizados de la etapa 1, pasaron a conformar nuevas suites en la etapa 2, en forma directa o realizándoles pequeños cambios. Cabe aclarar que en este proyecto, la ejecución de dichas suites y los incidentes encontrados por las mismas con cada nueva versión del producto no son parte de las tareas realizadas por el equipo de pruebas del CES, esa tarea la realizaba directamente el cliente.

NÚMEROS DEL PROYECTO		
	Etapa 1 20/11/06 al 18/01/07	Etapa 2 11/03/07 al 03/06/07
Suites	4	26
Scripts	46	271
% Scripts reutilizados etapa anterior	0%	40%
% Scripts nuevos	100%	60%

Tabla 2 – Suites y Scripts generados por etapa

A continuación se describe la experiencia en el uso de la metodología de la sección 2.1 en conjunto con las herramientas de la sección 2.2 en los proyectos de automatización llevados a cabo.

El primer paso en un proyecto de automatización de pruebas funcional es definir aquellas funcionalidades o ciclos funcionales que resulta importante o beneficioso automatizar. También evaluar cuales son los más fáciles de automatizar o los más ejecutados. Esto constituye la actividad AT1 – “Definición de las pruebas automatizadas”.

Una vez definidas las funcionalidades, se debe escribir los procedimientos de prueba en los cuales se definen las acciones y verificaciones que conforman las pruebas.

En este paso, es bueno tener en mente las posibilidades que ofrece Selenium en cuanto a las acciones y verificaciones posibles. Selenium permite ejecutar código Javascript arbitrario, por lo tanto, pone a disposición para estos propósitos todo el potencial de este lenguaje.

En el inicio, se comenzó utilizando un documento que especificaba los ciclos funcionales con las funcionalidades que involucra junto con una planilla electrónica donde se enumeran los pasos a seguir en cada pantalla. Luego se notó que era muy costoso llegar a este nivel de detalle y no proporcionaba un beneficio considerable. Se optó por un documento donde se detallan los ciclos funcionales y para cada funcionalidad se especifica las verificaciones que se deben realizar. Los ciclos funcionales y las pruebas de las funcionalidades se transformarán luego en suites y scripts respectivamente. Esto forma parte de la actividad AT2 – “Definición de los procedimientos de prueba”.

Una vez definidas las pruebas se puede comenzar a crear y documentar el entorno de datos, esto es AT6 – “Configuración de entorno”. Las actividades llevadas a cabo hasta el momento no requieren herramientas más allá de un editor de texto y una planilla electrónica.

La actividad AT3 – “Generación de suites y scripts” comienza cuando están las pruebas definidas y el entorno de datos configurado. En ella los testers encargados de la automatización utilizan Selenium IDE que permite grabar y editar script fácilmente. Esta herramienta, como ya dijimos, es una extensión de Mozilla Firefox. Las extensiones para este navegador se convierten en una herramienta fundamental para el tester automatizador. Si la aplicación no soporta este navegador, los scripts no podrán ser grabados sino que se deberán codificar en forma manual.

Una vez grabados o codificados, los scripts son probados y, en ocasiones, ajustados para que funcionen en otros navegadores. Cabe destacar que el tester requiere aprender sobre la herramienta de automatización Selenium Core, la herramienta de grabación Selenium IDE, el lenguaje HTML-Selenese (de Selenium Core), estándares de programación y procedimientos de gestión de las pruebas definidos por el equipo de automatización. A medida que el desarrollador va adquiriendo experiencia va dejando de lado herramientas como el grabador Selenium IDE y comienza a escribir directamente los script a través de un editor de texto. En nuestro caso utilizamos el entorno de desarrollo Eclipse.
Los scripts se deben mantener bajo gestión de configuración. Para ello se utiliza un servidor CVS donde se colocan todos los artefactos generados en el proceso. Como cliente CVS usamos el propio Eclipse.
Una vez construido el script, se procede a reproducirlo en el servidor Web a través de Selenium Core. Este es un proceso iterativo de ajuste y prueba.
Luego de construidas las suites se procede a ejecutarlas en el orden preestablecido de manera de verificar su correcto funcionamiento y verificar las dependencias. Esto requiere ejecutar secuencialmente en Selenium Core cada una de las suites observando su comportamiento. Esto es parte de AT4 – "Ejecución de las pruebas automatizadas del ciclo funcional".
Luego, se procede con la actividad AT7 – "Validación de las pruebas automatizadas" ejecutando las suites en el ambiente del cliente y verificando el correcto funcionamiento. De esta manera el cliente puede comenzar la validación de las pruebas. Cuando el mismo acepta las pruebas se cierra el proceso.
La actividad AT8 – "Organización de las pruebas automatizadas" se ejecuta en paralelo y durante todo el proyecto. En ella se deben definir, mantener y actualizar los procedimientos de gestión de las pruebas. No se contó con una herramienta que facilite la tarea.
Resulta fundamental la actividad AT5 – "Investigación y modificación de herramientas" pues una de las características más importantes de Selenium es tener a disposición el código fuente. Las posibilidades que ofrece Selenium son amplias (comandos y ejecución de código Javascript del usuario a través del comando *"waitForCondition"*), sin embargo, puede ser necesario realizar extensiones de manera de ajustarnos a características particulares de la aplicación bajo prueba.
Desde que comenzamos a usar Selenium hasta la fecha hemos implementado varias extensiones que nos permitieron nuevas posibilidades en el manejo de ventanas emergentes, soluciones para subir archivos desde el cliente al servidor Web e implementaciones de Selenium Core que ejecutan las pruebas a velocidades diferentes. Los foros de OpenQA son una fuente importante de información.

2.4 Conclusiones de la automatización

De nuestra experiencia práctica en el último año, podemos asegurar la factibilidad de proyectos de automatización de pruebas funcionales para aplicaciones Web utilizando herramientas Open Source.
Los clientes se mostraron satisfechos con el producto resultante de la automatización: scripts, suites y documentación. En los proyectos en lo que hemos participado, los scripts quedaron en propiedad del cliente y son utilizados para ejecutar las pruebas automatizadas sin la necesidad de personal del CES.
En cuanto a Selenium, demostró ser simple, potente y flexible, además de proveer un lenguaje fácil de usar y de aprender. La documentación de Selenium disponible en el sitio se limita a una referencia de comandos, por lo tanto los foros son una importante fuente de información. En general los usuarios de Selenium tienen gran participación en los foros y hay muchas extensiones disponibles de la herramienta.
También debemos destacar algunos aspectos negativos en el uso de Selenium Core, por ejemplo, no se puede obtener datos de fuentes externas como puede ser una base de datos. Sin embargo, si se

utiliza Selenium Remote Control no tenemos este inconveniente, ya que se tiene a disposición todas las posibilidades del lenguaje de programación en el cual se escriben las pruebas. Hay otras dos limitaciones que tiene Selenium Core y que, al igual que en el caso anterior, se pueden sortear utilizando Selenium Remote Control. La primera es que los scripts no manejan instrucciones o comandos de bifurcación, esto es una decisión de diseño de la herramienta, en busca de simplicidad. La segunda es que Selenium Core debe estar instalado en el mismo servidor Web que la aplicación a probar.
Uno de los principales problemas al automatizar aplicaciones utilizando el conjunto de herramientas open source indicadas se encuentra en la administración las suites y scripts resultantes de forma adecuada. Como se muestra en la Tabla 2, en un proyecto de 5 meses de duración se generaron más de 300 scripts, los cuales deben ser organizados y mantenidos. El costo de la administración de los artefactos resultantes de las pruebas no es para nada despreciable. Las tareas relativas a esta área del proyecto se llevaron a cabo manualmente por el equipo de automatización, pero se sintió la necesidad de una herramienta informática que brinde cierta asistencia. Interesa encontrar una herramienta open source que, entre otras cosas, permita administrar los artefactos generados y mantener trazabilidad entre las suites y scripts y las funcionalidades que estas prueban.

3 PROPUESTA PARA LA GESTIÓN DE LAS PRUEBAS AUTOMATIZADAS

A partir de la necesidad de una herramienta que permita gestionar los artefactos generados en la automatización, se investigaron diferentes alternativas para lograr gestionarlos de forma eficiente. Se propone la conjunción de la herramienta FitNesse y Selenium Remote Control para crear y gestionar suites, scripts y documentación de una forma adecuada.
En la sección 3.1 se resume la herramienta FitNesse. En la sección 3.2 se presenta la propuesta para la gestión de las pruebas y en la sección 3.3 se describe el trabajo a futuro para validar dicha propuesta.

3.1 FitNesse

FitNesse es una herramienta para mejorar la comunicación y colaboración en un proyecto de desarrollo de software. Fue creado por Robert Martin, Micah Martin y Michael Feathers, parte del equipo de Object Mentor [19]. El proyecto sigue en desarrollo y utiliza la licencia GNU General Public License versión 2.
FitNesse brinda un espacio donde documentar lo que el sistema debe hacer y compararlo con lo que el sistema hace. En este espacio pueden colaborar cliente, programadores y testers desde el inicio del desarrollo. El cliente puede generar sus propias pruebas, así como ejecutar las pruebas que escribieron los diseñadores de pruebas.
FitNesse es una wiki, lo que aporta facilidad para crear y editar páginas web. Las pruebas se definen mediante tablas dentro de las páginas. Estas pruebas se pueden ejecutar y FitNesse comparará los resultados esperados con los resultados obtenidos.
FitNesse provee interfaces a ser implementadas, mediante los cuales ejecuta una prueba y en caso de ser necesario compara el resultado esperado con el obtenido. Estas interfaces deben ser implementadas en Java.

3.2 Gestión de las Pruebas Automatizadas con FitNesse

Se propone utilizar Selenium Remote Control en Java para implementar las interfaces provistas por FitNesse. Con esto se podrían definir, por ejemplo, los comandos de Selenium Core para luego utilizarlos en los scrips ingresados en las páginas de FitNesse. También crear comandos más sencillos que encapsulen el código JavaScript, logrando así que el diseñador de pruebas se enfoque

en las técnicas de prueba sin preocuparse de aspectos de tecnología y el tester automatizador en diseñar e implementar la interfaz de FitNesse. Esto aumentaría la calidad de las pruebas como también la productividad de la automatización.
Al tratarse de una wiki se podría aprovechar el potencial en cuanto a la gestión de sus páginas. Dentro de las ventajas que ofrece un sistema wiki se encuentran el versionado de las páginas y la asistencia al encontrar inconsistencias en páginas por edición simultánea.

3.3 Trabajo a Futuro

Hasta el momento nuestro trabajo se basó en el uso de la herramienta de automatización Selenium Core y actualmente estamos abordando el trabajo con Selenium Remote Control integrado a FitNesse. A futuro pretendemos implementar suites y scripts más complejos para conocer a fondo las fortalezas y debilidades de estas herramientas integradas.
El objetivo de integrar estas herramientas es crear una interfaz sencilla que permita ejecutar comandos similares a Selenium Core. Para cada aplicación a probar se podría extender la interfaz tomada como base, de manera de definir nuevos comandos y enriquecer el lenguaje. Puede ser útil la colaboración del cliente en la definición del lenguaje aportando comandos que se adapten a sus necesidades.
Otro aspecto interesante que podría surgir en el uso de FitNesse al utilizar Selenium Remote Control con Java es la posibilidad de utilizar todo el potencial que nos provee Java para acceder a bases de datos. De esta forma es posible verificar, por ejemplo, que la aplicación bajo prueba se comporte de la forma esperada consultando para ello la base de datos. Este enfoque no es posible con Selenium Core debido a que permite automatizar pruebas que verifican la aplicación únicamente a través de la interfaz gráfica.
Cabe destacar que el uso de Selenium Remote Control en FitNesse provee una visualización de cómo se van comportando los scripts en su ejecución, característica que no esta presente al utilizar únicamente Selenium Remote Control pues no posee interfaz gráfica para ejecución de las pruebas.

4 CONCLUSIONES

Se presentó una metodología para la automatización de las pruebas funcionales automatizadas y un conjunto de herramientas open source que asisten a dichas actividades.
De nuestra experiencia práctica en el último año, podemos asegurar la factibilidad de proyectos de automatización de pruebas funcionales para aplicaciones Web utilizando este conjunto de herramientas open source. Además, los clientes se mostraron satisfechos con el producto resultante de la automatización: scripts, suites y documentación. Las suites y scripts generados son utilizados para ejecutar las pruebas automatizadas sin la necesidad de personal del CES.
En cuanto a Selenium, demostró ser simple, potente y flexible, además de proveer un lenguaje fácil de usar y de aprender.
Uno de los principales problemas al automatizar aplicaciones utilizando el conjunto de herramientas open source presentadas en este artículo se encuentra en la gestión de los artefactos generados en el proyecto. Como trabajo a futuro, se propone la conjunción de la herramienta Fitnesse y Selenium Remote Control para crear y gestionar suites, scripts y documentación de una forma adecuada.

REFERENCIAS

[1] Apache License 2.0 - http://www.apache.org/licenses/LICENSE-2.0

[2] Beizer B. “Software testing techniques (2nd ed.)”, ISBN:0-442-20672-0, Van Nostrand Reinhold Co, 1990.

[3] Black R. “Managing the Testing Process, 2nd Edition”. ISBN 0-471-22398-0, Editorial Wiley, 2002.

[4] Camara Uruguaya de tecnologías de información (CUTI). http://www.cuti.org.uy/

[5] Centro de Ensayos de Software (CES) .http://www.ces.com.uy, 2007.

[6] Concurrent Version System. http://www.nongnu.org/cvs/

[7] Dustin E., Rasca J., Paul J. “Automated Software Testing”, ISBN 0-201-43287-0, Addison Wesley, 1999.

[8] Eclipse. http://www.eclipse.org/

[9] Firebug - http://www.getfirebug.com

[10] FitNesse - http://FitNesse.org/

[11] Genexus – http://www.genexus.com

[12] IEEE Standard Glossary of Software Engineering Terminology Institute of Electrical and Electronics Engineers,ISBN: 155937067X, 1990.

[13] International Software Testing Qualifications Board, Certified Tester Foundation Level Syllabus, Versión 2005. http://www.istqb.org/fileadmin/media/SyllabusFoundation.pdf

[14] Internet Explorer. http://www.microsoft.com/spain/windows/products/winfamily/ie/default.mspx

[15] Kaner C., Bach J., Pretichord B. “Lessons Learned in Software Testing”, ISBN 0471081124, Wiley, 2001.

[16] Kaner C., Falk J., Nguyen H. “Testing Computer Software, 2nd Edition”, ISBN: 0471358460, Wiley, 1999 .

[17] Mozilla Firefox. http://www.mozilla-europe.org/es/products/firefox/

[18] Myers G. “The art of software testing, 2nd edition”, ISBN 0-471-46912-2, John Wiley & Sons Inc., 2004.

[19] Object Mentor - http://www.objectmentor.com/

[20] OpenQA Sitio Web http://www.openqa.org/

[21] Pérez B., “Proceso de Testing Funcional Independiente (ProTest)”, Tesis de Maestría en Informática, PEDECIBA Informática, Facultad de Ingeniería, Universidad de la Republica, Uruguay, ISSN: 0797-6410 - 06-11, 2006.

[22] Selenium. http://www.openqa.org/selenium/

[23] Selenium Core. http://www.openqa.org/selenium-core/

[24] Selenium Ide. http://www.openqa.org/selenium-ide/

[25] Selenium Remote Control. http://www.openqa.org/selenium-rc/

[26] Software Test Automation, Effective use of test execution tools. Mark Fewster & Dorothy Graham. ISBN 0-201-33140-3

[27] ThoughtWorks. http://www.thoughtworks.com/

[28] Universidad de la República (UdelaR). http://www.universidad.edu.uy/index.php

[29] XPath Checker - https://addons.mozilla.org/en-US/firefox/addon/1095

[30] Xpather - http://xpath.alephzarro.com