Evaluación de herramientas Free/Open Source para pruebas de software

Francisco Javier Díaz[1], Claudia M. Banchoff T.[1], Anahí S. Rodríguez[1] y Valeria Soria[1],
[1] Laboratorio de Investigación de Nuevas Tecnologías Informáticas, Facultad de Informática, Universidad de La Plata, Buenos Aires, Argentina.
{javier.diaz, claudia.banchoff, anahi.rodriguez, valeria.soria}@linti.unlp.edu.ar

Resumen

Todas las etapas en el proceso del desarrollo de software son sumamente relevantes, pero, quizás la etapa de pruebas o testing sea la menos sistematizada y tenida en cuenta en ese proceso. Existen numerosas herramientas, muchas de ellas Free/Open Source, orientadas a facilitar las actividades al grupo que realiza las pruebas, pero la incorporación de las mismas es en sí misma una tarea adicional que involucra no sólo el uso sino también la evaluación de muchas herramientas que ocupan el poco tiempo destinado a esta actividad.
Esta línea de investigación se basa en el campo de testing de software y el objetivo de este trabajo es plantear distintos criterios para la evaluación de herramientas Free/Open Source. Si bien estos criterios pueden ser utilizados en la evaluación de cualquier tipo de herramienta, se tendrá un foco especial en las destinadas a las actividades de pruebas de software.

Keywords: Pruebas/Testing, Free/Open Source, modelo de madurez.

1 Contexto

La línea de investigación presentada está inserta en el proyecto de incentivos del LINTI "*Redes, Seguridad y Desarrollo de Aplicaciones para e-educación, e-salud, e-gobierno y e-inclusión*".
Este trabajo es uno de los puntos abordados en la tesina de grado de fin de carrera de la Licenciatura en Sistemas de la Facultad de Informática de la UNLP, de las alumnas Anahí Rodríguez y Valeria Soria, dirigidas por Lic. Claudia Banchoff y Lic. Javier Díaz.

2 Introducción

En el proceso de desarrollo de software, las actividades relacionadas con la etapa de testing suelen llevarse a cabo luego de finalizada la implementación, y, en muchos casos, terminan realizándose en paralelo con puestas en producción. Esto trae las obvias consecuencias de sistemas de actualizaciones y parches constantes que permiten reparar problemas que podrían haber sido salvados con una mejor revisión.

Si bien existen muchas herramientas que permiten asistir en la etapa de testing (que no siempre se deben aplicar como etapa final del desarrollo), la elección de las mismas es una tarea adicional que debe realizar el equipo. Esto se agrava en el caso de querer trabajar con herramientas Free/Open Source, donde se deben considerar otras cuestiones propias del modelo de desarrollo de dichos software. [1]

En este artículo, se van a plantear distintos modelos de evaluación de herramientas Free/Open Source, haciendo foco en aquellas que permiten asistir al equipo de pruebas en sus tareas.

En las siguientes secciones se describirán los distintos modelos analizados, concluyendo al final del artículo con una propuesta que combina distintos aspectos de varias de estas metodologías.

3 Metodologías de evaluación de proyectos Free/Open Source

Para construir un modelo de selección de herramientas Free/Open Source para el desarrollo de tareas de testing, se tomaron como base mecanismos existentes, que permiten evaluar la madurez de un producto Free/Open Source.

El modelo de madurez del proceso de desarrollo de un software evalúa el nivel de calidad con CMMI [2]. Este modelo es demasiado complejo y la infraestructura del proceso que utiliza no es del todo aplicable a los procesos de desarrollo de software Free/Open Source.

Es por esto que, en los últimos años, se han desarrollado varios modelos para definir un proceso de evaluación de software Free/Open Source. Algunos de estos se centran en ciertos aspectos como la madurez, la durabilidad y la estrategia de la organización en torno al proyecto Free/Open Source en sí. Otros modelos agregan aspectos funcionales al proceso de evaluación.

A la hora de seleccionar un software Free/Open Source, es recomendable contar con un método de clasificación, examinando las limitaciones y los riesgos que tiene este tipo de software.

Algunas preguntas generales para hacer antes de seleccionar un software son las siguientes:

- ¿Cuál es la durabilidad del software?
- ¿Qué nivel de estabilidad tiene o se espera?
- ¿Es fácil y factible agregarle funcionalidad al software?

- ¿Tiene una comunidad activa que lo avale?

Es por esto que se definieron los modelos de madurez de proyectos de software Free/Open Source, los cuales en su mayoría cuentan con:

- Una puntuación según las características del software.
- Una elección final, según los criterios de puntuación establecidos.

[3]

A continuación se resumen los distintos modelos analizados que fueron tomados como base de la metodología propuesta:

3.1 OSMM (Open Source Maturity Model de Capgemini) [4] [5]

Criterios planteados: Define 27 parámetros a analizar entre los cuales, 12 de ellos están divididos en 4 categorías genéricas del producto (Producto, Integración, Uso y Aceptación), por ejemplo: edad (Inicio del Proyecto), licencia, comunidad de desarrollo, colaboración con otros productos, cumplimiento de estándares, etc. y los 15 restantes son características basadas en las necesidades del usuario, por ejemplo: usabilidad, seguridad, interface, independencia de la plataforma, soporte, etc.

Metodología: Cada categoría es analizada por lo menos por dos evaluadores que asignan un puntaje entre 1 (menos importante) y 5 (más importante) que determinarán la madurez del producto.

3.2 OSMM (Open Source Maturity Model de Navica) [6] [7]

Criterios planteados: Se tienen en cuenta 6 características: Software, Soporte, Documentación, Capacitación, Integración y Servicios de profesionales. Este modelo propone una plantilla por cada una de ellas, donde se sugieren diferentes aspectos a ser evaluados. También define algunos elementos que permiten una evaluación parcial del proceso de desarrollo.

Metodología: Este proceso consta de 4 fases:

1) Seleccionar el software que se va a evaluar.

2) Ponderar cada una de las características.

3) Aplicar las plantillas correspondientes a las categorías y asignarles un puntaje a cada uno de los aspectos que integran las categorías.

4) Multiplicar la puntuación de cada categoría por su ponderación para producir una puntuación final de 0 a 100.

3.3 QSOS (Qualification and Selection of Open Source Software) [8]

Criterios planteados: Este modelo permite identificar si el software cumple los requisitos técnicos, funcionales y estratégicos, mediante la clasificación y comparación de los diferentes productos según los criterios ponderados, con el fin de tomar una decisión final. Para ello provee plantillas y cuadros para poder clasificar y evaluar los diferentes productos.

Metodología: El proceso consiste en 4 pasos (siendo independientes entre si e iterativos):

1) Definición: se clasifica el software, la licencia y la comunidad a la que pertenece.

2) Evaluación: Se evalúa el software en 3 ejes: cobertura funcional, riesgo del usuario y los riesgos de los desarrolladores.

3) Calificación: Se definen los filtros en base a los puntos que se evaluarán. Sirve para eliminar el software que no satisface las necesidades del usuario.

4) Selección: Aplicación del paso 3, con los datos dados por los primeros pasos, formulando consultas, comparaciones y la selección del producto.

3.4 BRR (Business Readiness Rating) [9]

Criterios planteados: Define 12 criterios que pueden ser utilizados para evaluar el software una vez que se estableció la lista inicial de productos. Se sugiere que sólo 6 o 7 criterios sean realmente utilizados en la evaluación. Algunos de estos criterios son por ejemplo: funcionalidad, usabilidad, calidad, seguridad, rendimiento, escalabilidad, arquitectura, soporte, documentación, adopción, comunidad y profesionalismo.

Metodología: El modelo de evaluación incluye 4 pasos:

1) Se crea una lista con el software a evaluar.

2) Se clasifica y se ponderan los criterios de selección.

3) Se recopilan datos para cada criterio.

4) Se calculan y se publican los resultados.

3.5 NASA's Reuse Readiness Levels (RRL) [10]

Criterios planteados: WG (Software Reuse Working Group) identificó 9 áreas temáticas que se consideran importantes para medir la madurez en la reutilización del software, como ser documentación (guías de instalación, manual de usuario, foros, tutoriales, etc.), extensibilidad, cuestiones de propiedad intelectual, modularidad, empaquetado, portabilidad, cumplimiento de estándares, soporte (Apoyo técnico, comunidad de usuarios, ayuda online, etc.) y verificación y pruebas.

Siguiendo el formato de TRL (Technology Readiness Levels), existen 9 niveles de RRL (Levels of Preparation for Reuse) que van de 1 (menos maduro) a 9 (más maduro).

Metodología: Este método define una tabla (producto cartesiano) donde se tienen las 9 áreas junto a los 9 niveles con una breve descripción (Reuse Readiness Level (RRL) Topic Area Levels Summary). RRL proporciona una guía para la reutilización, reduciendo el número de posibles soluciones.

4 Modelo de evaluación propuesto

Tomando como base de conocimiento los métodos analizados en las secciones anteriores, se propone un modelo que sintetice aquellos aspectos o criterios más relevantes. Estos criterios de evaluación no están basados en un método específico, dado que el objetivo de la evaluación que se va a realizar no es determinar la madurez del software sino seleccionar, bajo ciertos criterios, un subconjunto de herramientas del total existente. [3]

Los puntos más importantes a ser evaluados se muestran en la siguiente tabla:

Descripción general	Se describe brevemente la funcionalidad de la herramienta, mencionando la URL del sitio oficial, el lenguaje en el que está desarrollada, etc.
Documentación	Se detalla si la herramienta cuenta con guía de instalación, manual de usuario, FAQs, soporte online (foro, blog, lista de e-mails, etc.), si el código se encuentra comentado (Opcional).
Aspectos de Madurez	Se especifica la fecha de inicio del proyecto, cuál es la última versión, se cuenta con una puntuación para ponderar el grado de actualización, la actividad en lanzamientos, la actividad en el reporte de errores.
Licencia	Se nombra el tipo de licencia a la que pertenece la herramienta.
Plataforma	Se detalla el sistema operativo sobre el cual se puede instalar la herramienta (portabilidad)
Interfaz	Se informa el tipo de interfaz (GUI o consola) con el que cuenta la herramienta
Extensibilidad	Se verifica si el sitio oficial provee metodologías o procedimientos de desarrollo definidos, o accesos al CVS o SVN.

4.1 Metodología de evaluación

El modelo propuesto, consta des tres (3) fases.

Fase 1: se elige un subconjunto del total de herramientas existentes, que cumplan con un foco específico, y se completan los siguientes datos para cada una de ellas:

Nombre:
Descripción:
URL:

Fase 2: se completan las planillas de evaluación de acuerdo a los criterios que se mencionaron en el apartado anterior, con información proporcionada por el sitio oficial de la herramienta o sitios reconocidos.

Datos generales

Herramienta	Versión	Inicio del proyecto	Licencia	Plataforma	Interfaz	Lenguaje

Criterios de documentación

Herramienta	Guía de instalación	Manual de usuario	Preguntas Frecuentes	Soporte Online			Código comentado	Adicional
				Foro	Lista de mail	Blog		

Criterios de madurez (1)

Herramienta	Inicio del proyecto (2)	Grado de actualización (3)	Actividad en lanzamientos (4)	Actividad en el reporte de errores (5)

(1) Se completa con las siguientes siglas:
M = Malo B = Bueno
R = Regular MB = Muy bueno
(2) Se refiere a que si es muy joven quizás no tenga una madurez suficiente
(3) La última versión se encuentra cercana al año actual
(4) Se refiere a la frecuencia en la publicación de versiones
(5) Se refiere a la frecuencia con que se resuelven y reportan errores

Fase 3: se evalúa la información obtenida para la selección de la herramienta a utilizar.

4.2 Caso de uso

A continuación se muestra un ejemplo de uso, habiendo evaluado 3 herramientas para pruebas de rendimiento:

JMeter: Es una aplicación de escritorio desarrollada en JAVA, diseñada para medir el rendimiento y el comportamiento de los sistemas ante las pruebas de sobrecarga.
Se puede utilizar para probar el rendimiento tanto de los recursos estáticos como dinámicos (archivos, Servlets, scripts de Perl, Java Objects, bases de datos y consultas, servidores FTP y más). Puede ser utilizado para simular una sobrecarga en un servidor, una red o un objeto, para poner a prueba su resistencia o para analizar el rendimiento global para diferentes tipos de carga. Puede usarse para hacer un análisis gráfico de rendimiento o para probar su servidor / script / comportamiento del objeto con sobrecargas concurrentes.
URL: http://jakarta.apache.org/jmeter/

OpenSTA: Es un conjunto de herramientas que tiene la capacidad de realizar secuencias de comandos HTTP y HTTPS para pruebas de sobrecarga, para medir el rendimiento de aplicaciones en plataformas Win32. Permite captar las peticiones del usuario generadas en un navegador Web, luego guardarlas, y poder editar para su posterior uso.
URL: http://opensta.org/

WebLoad: Permite realizar pruebas de rendimiento, a través de un entorno gráfico en el cual se pueden desarrollar, grabar y editar script de pruebas.
URL: http://www.webload.org/

Datos generales

Herramienta	Versión	Inicio del proyecto	Licencia	Plataforma	Interfaz	Lenguaje
JMeter	JMeter 2.4 – Julio 2010 (visto en Octubre 2010)	Diciembre 1998	Apache License (Version 2.0, January 2004) Apache License es compatible con GPL	Independiente	GUI	Java
OpenSTA	OpenSTA 1.4.4 – Octubre 2007 (visto en Octubre 2010)	Febrero 2001	GNU GPL	Windows	GUI	C++
WebLoad	WebLoad 8.1.0 - Octubre 2007 (visto en Octubre 2010)	Febrero 2007	GPL y Professional (no es libre)	Windows	GUI	C++, Java

Criterios de documentación

Herramienta	Guía de instalación	Manual de usuario	Preguntas Frecuentes	Soporte Online			Código comentado	Adicional
				Foro	Lista de mail	Blog		
JMeter	NO	SI	SI	SI	SI	NO		Wiki Javadocs (API)
OpenSTA	NO	SI	SI	NO	SI	NO		
WebLoad	NO	NO	NO	NO	SI	NO		

Criterios de madurez

Herramienta	Inicio del proyecto	Grado de actualización	Actividad en lanzamientos	Actividad en el reporte de errores
JMeter	B	B	B	MB
OpenSTA	B	R	R	R
WebLoad	R	M	M	N/A

5 Conclusiones

Dado que existen miles de proyectos Free/Open Source, y que pueden estar en diferentes etapas de desarrollo (proyectos que han sido abandonados, que se encuentran en etapas tempranas de desarrollo, que alcanzaron un nivel estable para su utilización, etc.), se han desarrollado diferentes modelos que ayudan a determinar la madurez de los mismos (OSMM, QSOS, BRR, etc.).

Nuestro trabajo se basó en estos modelos, tomando los puntos más relevantes (a nuestro criterio) y confeccionando tres tablas para la selección de herramientas Free Open/Source. Se generó este nuevo modelo de evaluación ya que se ajusta más a nuestras necesidades, permitiéndonos enfocarnos en los puntos más importantes a verificar de las herramientas y no tanto en su madurez.

Este trabajo constituye un aporte sustancial al equipo de testing del LINTI/CeSPI, dado que se utiliza esta metodología para facilitar la selección de herramientas para esta área.

Como trabajo a futuro se propone evaluar y seleccionar herramientas para los diferentes tipos de pruebas y para distintos tipos de aplicaciones (Web o GUI). Cabe destacar que este modelo se puede aplicar también para la selección de cualquier tipo de herramienta.

6 Formación de Recursos Humanos

El presente trabajo muestra una línea de investigación que se inició en el año 2007. Dos investigadoras en formación se han capacitado en diferentes cursos, se han realizado distintos trabajos en el área de testing, se realizaron publicaciones en diferentes congresos, se ha impartido su conocimiento a otros grupos pertenecientes a ambos centros de investigación y se están formando para la realización de la tesina de grado.

Referencias

1. “Herramientas open source para testing de aplicaciones Web. Evaluación y usos.pdf” – J. Díaz, C. Banchoff, A. Rodríguez, V. Soria (2009) – CACIC 2009.
2. “Introducing the OpenSource Maturity Model” – Petrinja, E.; Nambakam, R.; Sillitti, A. – Junio 2009 – ISBN 978-1-4244-3720-7 – IEEE.
3. “No todo es codificar – Modelos de Madurez en SL” – 8vas Jornadas de Software Libre – Facultad de Informática – UNLP – A. Rodríguez, V. Soria
4. http://pascal.case.unibz.it/retrieve/1097/GB_Expert_Letter_Open_Source_Maturity_Model_1.5.31.pdf
5. http://www.osspartner.com/portail/sections/accueil-public/evaluation-osmm
6. http://www.oss-watch.ac.uk/resources/osmm.xml
7. http://web.archive.org/web/20080507024544/http://www.navicasoft.com/pages/osmm.htm
8. http://www.qsos.org
9. http://www.oss-watch.ac.uk/resources/brr.xml
10. http://www.esdswg.org/softwarereuse/Resources/rrls/RRLs_v1.0.pdf