Herramientas open source para testing de aplicaciones Web. Evaluación y usos

Javier Díaz[1], Claudia Banchoff Tzancoff[1], Anahí 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. Las aplicaciones informáticas han evolucionado con el correr de los años. Desde las aplicaciones de consola a las aplicaciones Web. Estas últimas, muy utilizadas hoy en día por la gran popularidad de Internet. Todas las etapas en el proceso del desarrollo de software son sumamente relevantes, pero, quizás la etapa de testing sea la menos sistematizada y tenida en cuenta. Prueba de esto, es la gran cantidad de parches y versiones surgidas luego del lanzamiento de una versión final de un software, destinadas a cubrir "agujeros de seguridad" o simplemente, malos funcionamientos. Por este motivo, es muy importante realizar pruebas de software que eviten sistemas de baja calidad y aumenten la confianza de los usuarios. El objetivo de este trabajo es presentar una serie de herramientas que asisten en la etapa de testing de un sistema. Para este trabajo se ha focalizado en el análisis de aplicaciones Web. Se presentarán herramientas de código abierto agrupadas según el tipo de prueba realizar.

Palabras Claves: Testing - software libre – calidad.

1 Introducción

Las aplicaciones informáticas han evolucionado con el correr de los años. Desde las aplicaciones de consola a las aplicaciones Web, pasando por las aplicaciones de escritorio con interfaz gráfica.

En lo que se refiere aplicaciones Web, las mismas han evolucionado notablemente desde su aparición. En un principio se podía hablar de sitios Web compuestos por páginas estáticas (sólo texto e imágenes) y luego fueron evolucionando hasta lo que conocemos hoy en día, donde existen distintos tipos de aplicaciones Web como correo, foros, chats, buscadores, documentos on-line, CMS (Sistemas de Gestión de Contenidos), sistemas que dan soporte a las redes sociales, tiendas virtuales, mapas on-line, juegos, etc.

Según Myers [1]: “Las aplicaciones de Internet son aplicaciones cliente-servidor donde el cliente es un navegador Web y el servidor es un servidor Web o una aplicación servidor".

El desarrollo, las estrategias de interfaz de usuario v también el enfoque de las pruebas varían para los diferentes tipos de sitios Web.

Al igual que el desarrollo de software, las pruebas también tienen diferentes etapas, como ser: Planificación y Control, Análisis y Diseño, Implementación y Ejecución, Evaluación y Cierre.

Las pruebas del software pueden realizarse en distintas etapas del proceso de desarrollo. Es importante que las mismas se realicen en etapas tempranas, pudiendo esto, obviamente, condicionar el posterior desarrollo. Si se realizan en etapas tempranas es posible mejorar la calidad del producto, es menos costoso encontrar errores y resolverlos en las primeras etapas que al final del desarrollo (se deben encontrar errores antes de que los encuentre el cliente).

Los errores en el software pueden tener consecuencias drásticas si no son tratados a tiempo. Existen muchos ejemplos de esto[37] y en todas estas situaciones un buen análisis en el momento adecuado podrían haber evitado estas situaciones.

Si bien, hay pruebas que deben ser realizadas en forma manual, existen numerosas herramientas, tanto privativas como de código abierto, que asisten al equipo de testing. La utilización de estas herramientas facilita la ejecución de las pruebas pero no garantiza el éxito en su uso. En muchos casos, el tiempo de aprendizaje de las mismas o la correcta selección implica un esfuerzo adicional que no en todos los proyectos puede ser tenido en cuenta. Muchas herramientas están focalizadas en un tipo de prueba específica y otras son más generales. Una correcta elección no es una tarea sencilla.

En las siguientes secciones de este artículo se presentan aspectos importantes de los distintos tipos de pruebas, y un análisis comparativo de herramientas de código abierto agrupadas según el tipo de prueba que se puede realizar en una aplicación Web.

2 La Calidad del Software

Con las pruebas de cualquier desarrollo de software se pretende asegurar mayor calidad al producto.

El concepto de calidad de software es un concepto muy distinto al de cualquier producto o servicio del sector industrial, ya que por ejemplo la IEEE std. 610, lo define como “El conjunto de programas de ordenador, los procedimientos y posiblemente, la documentación asociada y los datos relativos a la operación del sistema informático”, y tiene unas características muy especiales ya que se desarrolla y no se fabrica; es un producto lógico y no físico; y no se degrada con el uso.

El concepto de calidad de software no es tan sencillo como parece, pues cuidar de la calidad del software significa cuidar todos y cada uno de los elementos enumerados en la definición del mismo y no sólo el código fuente.

Según los autores Olsina, Lafuente y Rossi [3], los valores de calidad que tiene mayor relevancia en las aplicaciones Web son los siguientes: Usabilidad, Funcionabilidad, Fiabilidad, Seguridad, Eficiencia y Mantenibilidad.

Aún así para determinar cuándo una aplicación es “buena” se debe tener en cuenta también la diversidad de usuarios que acceden a la misma, y así contar con una diversidad de opiniones por lo que pueden haber distintos puntos de vista para una misma Web [4].

3 Tipos de Pruebas

Para poder hablar de calidad del software es imprescindible asegurarnos que cada etapa de su desarrollo ha sido testeada acorde al tipo de aplicación y funcionalidad. Existen varios tipos de pruebas y el autor Myers [1] las clasifica en:

Pruebas de Aceptación: este tipo de pruebas se desprenden directamente de las especificaciones de los requerimientos del usuario, verificando que el software realiza lo especificado (reglas de negocio). Se describe un escenario de ejecución o uso del sistema desde la perspectiva del usuario, teniendo en cuenta requisitos funcionales o no funcionales. Cada uno de los requisitos puede tener una o más pruebas de aceptación. Algunas técnicas utilizadas son: diagramas de secuencia, casos de uso, bocetos de la UI, etc. [5].

Según Pressman [6], para evaluar los requerimientos de un usuario, podemos realizar pruebas alfa[1] o beta[2], que sirven para que el usuario descubra errores, que sólo él puede identificar como errores.

Pruebas de Sistemas: En este caso son varias las características a evaluar y analizar.

a) Usabilidad: Según la ISO/IEC 9126: "La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso". Por ejemplo: verificación de tamaño de los controles, redacción, colores, tipos de letra, etc.

b) Interfaz de usuario: Se refiere a la validación de datos, modos de ventanas (ubicación, tamaño, velocidad, etc.), menúes, etc.

c) Seguridad: Estas pruebas se basan tanto en la seguridad de los datos como también en la seguridad de la aplicación. En las aplicaciones Web hay mayores niveles de seguridad ya que se deben proteger las transacciones en Internet. Para mayor información consultar en OWASP [7].

d) Rendimiento: Se debe evaluar el grado con que un sistema o componente logra la funcionalidad señalada dentro de las restricciones dadas con respecto a tiempo de proceso. Comprobando el comportamiento del sistema ante determinadas situaciones, por ejemplo: cuanto tarda el servidor en responder ante el acceso de varios usuarios al mismo tiempo (Pruebas de Carga), peticiones de datos en casos extremos o resistencia ante el ingreso de grandes volúmenes de datos (Pruebas de Volumen) y verificar que el sistema sea estable mas allá de los limites especificados, permite verificar la robustez y escalabilidad de la infraestructura de la red (Pruebas de Stress).

e) Configuración: Se valida que el sistema funcione correctamente, por ejemplo para distintos tipos de hardware, en distintos sistemas operativos, para distintos manejadores de bases de datos, en el caso de aplicaciones Web, que funcionen en distintos navegadores Web y además en sus distintas versiones.

Otros tipos de prueba dentro de esta categoría pueden ser Fiabilidad, Compatibilidad, Instalación, entre otras.

Pruebas Funcionales: Las pruebas funcionales son aquellas que validan las especificaciones definidas por el usuario, teniendo en cuenta a la funcionalidad como

---

[1] Se realiza la prueba en el lugar del desarrollo en un ambiente controlado. El desarrollador observa los caminos realizados por el usuario, registrando los errores y problemas de uso.

[2] Se realiza la prueba en el lugar de trabajo del cliente, sin estar presente el desarrollador, en un entorno que no es controlado por el mismo.

una caja opaca en la cual se ingresan datos/valores y se debe controlar los valores/datos de salida. Existen distintas técnicas para poder implementar estas pruebas, como ser: técnicas por partición de equivalencia, valores de frontera, etc.
Pruebas de Integración: una vez realizadas las pruebas unitarias, se prueba la integración de cada una de las partes, esto se refiere a probar la interacción entre cada uno de los módulos. Existen dos técnicas top-down y bottom-up.
Pruebas de Unidad: este tipo de prueba se realiza cuando los programas son muy extensos, entonces se los modulariza, y se prueba cada una de estas partes, realizando más fácil las tareas de detectar los errores y poder corregirlos. Para este tipo de pruebas se realizan distintos casos de prueba para cada una de las funciones/procedimientos (o métodos).

4 Probando Aplicaciones Web

El tipo de pruebas no depende del tipo de aplicación, ya sea Web o escritorio. Como se mencionó en un comienzo, este trabajo está orientado al testeo de aplicaciones Web. En este caso, las pruebas básicas que se deben realizar, son: pruebas de aceptación, pruebas estáticas de código, pruebas unitarias, funcionales y de rendimiento.

Para Whittaker [8], la utilización de una herramienta puede ser útil para mejorar y agilizar las pruebas a realizar, por ejemplo, a la hora de simular una cantidad determinada de usuarios accediendo a un sitio al mismo tiempo, o la carga de datos en un sistema, etc. Aunque, en el momento del análisis de los resultados, es necesaria la presencia de un especialista que pueda interpretar y analizar los datos arrojados por la misma y así llegar a una conclusión.

El uso de una herramienta no garantiza el éxito en las pruebas, dado que para obtener resultados útiles es necesario conocer cómo utilizar estas herramientas y cómo interpretar sus resultados. También, en muchos casos, se requiere un esfuerzo adicional para lograr obtener beneficios de las mismas, como por ejemplo aprender cómo se utiliza, o si se necesita un conocimiento de un lenguaje propio de la herramienta para la generación de scripts de pruebas, etc.

El uso de herramientas en las pruebas, posee tanto beneficios como riesgos [9].

***Beneficios o Ventajas**:*

- *El trabajo repetitivo se reduce*: por ejemplo, en el caso de realizar pruebas de regresión en las cuales se deben volver a ejecutar los casos de pruebas ya realizados, o el reingreso de los datos al volver a cero o restaurar la aplicación.
- *Una mayor coherencia y repetibilidad*: al estar la prueba automatizada es mucho más cómodo y fácil de automatizar y no cometer errores.
- *Evaluación Objetiva*: las herramientas nos pueden proporcionar medidas estáticas o de cobertura.
- *Fácil visualización de los resultados*: algunas herramientas proporcionan gráficos, tablas, etc., los cuales muestran resultados.

***Riesgos o desventajas**:*

- Se pueden llegar a tener expectativas irrealistas sobre las herramientas: el uso de las mismas puede ser fácil/difícil, o también puede suceder que se crea que es para una funcionalidad en particular, y en realidad la herramienta fue creada para otro propósito.
- Sobrestimar el tiempo: el tiempo que puede llevar en la familiarización con la herramienta puede ser mucho más alto de lo que se había estimado, también puede llevar mucho tiempo tener experiencia en toda la funcionalidad que la herramienta provee.
- La incorporación de la herramienta a las pruebas realizadas puede demandar más tiempo que el previsto. (Tiempo de adaptación de la herramienta al ciclo de las pruebas)

4.1 Evaluación de Herramientas Según el Tipo de Prueba

En el presente trabajo se seleccionaron las siguientes herramientas para: pruebas de aceptación, estáticas de código, unitarias, funcionales y de rendimiento, en sitios Web de tipo informativo desarrollados en lenguaje Java y PHP. Existen varias herramientas similares, muchas propietarias, que no fueron analizadas en el presente artículo [10] [11] [12]. Se han seleccionado aquellas de código abierto y con más popularidad.

4.1.1 Herramientas Para Pruebas de Aceptación

Si bien este tipo de pruebas pueden realizarse con prototipos de la aplicación, diagramas de secuencia o casos de uso que se muestran al usuario para que valide los requerimientos definidos, existen algunas herramientas que pueden ser útiles:

FitNesse [13]: *P*ermite a los usuarios, equipos de testing y programadores aprender lo que debe hacer el software y comparar automáticamente lo que realmente hace. Se pueden realizar pruebas de aceptación y pruebas de reglas de negocio. Es una wiki que no requiere demasiadas configuraciones.

Avignon [14]: Es un framework para pruebas de aceptación que permite a los usuarios expresar pruebas de aceptación de una forma no ambigua antes que comience el desarrollo. Trabaja en conjunto con JUnit, HTTPUnit, JAXP y Xalan. Utiliza XML para definir la sintaxis del lenguaje.

4.1.2 Herramientas Para Pruebas Estáticas de Código

En este tipo de prueba es muy importante la utilización de herramientas, dado que las mismas facilitan las tareas del equipo de testing. Con su uso, es posible encontrar más fácilmente errores tales como: código muerto, código fuente no documentado o mal finalizado, etc. Se pueden utilizar las siguientes herramientas:

PHPLint [15]: Permite realizar pruebas de código fuente, con esto se pueden mejorar las tareas de programación, ya sea comenzando la codificación con esta herramienta o bien, mejorando código ya existente. Brinda buenas prácticas que permiten dar seguridad en el código, errores de sintaxis, variables no utilizadas, código muerto, etc.

RATS [16]**:** Realiza chequeo de seguridad en el código, determinando la criticidad de fallos, como así también una evaluación del código.
YASCA [17]: Permite encontrar vulnerabilidades de seguridad, calidad en el código, rendimiento, etc. Aprovecha la funcionalidad de los plugins FindBugs, PMD y Jlint.
PMD [18]: Puede ser integrado a varias herramientas: JDeveloper, Eclipse, JEdit, JBuilder, BlueJ, TextPad, Maven, Ant, Gel, Jcreator, etc. Permite encontrar en el código errores en el manejo de excepciones, código muerto, código sin optimizar, código duplicado, etc.
FindBugs [19]: Puede integrarse a Eclipse. Realiza un escaneo de código encontrando errores comunes, malas prácticas de programación, código vulnerable, rendimiento, seguridad, etc.

4.1.3 Herramientas Para Pruebas Unitarias

En este tipo de prueba el uso de herramientas facilita tareas como probar una clase o un módulo en particular. Las herramientas dependen del lenguaje utilizado. Algunas de ellas son:
JUnit [20]: Es un framework para la automatización de las pruebas unitarias y de integración. Provee clases y métodos que facilitan la tarea de realizar pruebas en el sistema y así asegurar la consistencia y funcionalidad.
PHPUnit [21]: Es un framework para PHP que permite crear y ejecutar tests unitarios de manera simple. Está basado en el framework "JUnit" para java.
SimpleTest [22]: Es un framework para pruebas de unidad en PHP y pruebas Web. Esta herramienta cuenta con un navegador Web interno, lo que permite que las pruebas naveguen los sitios Web, ingresen datos en formularios y páginas de prueba.

4.1.4 Herramientas Para Pruebas Funcionales

Para este tipo de pruebas, se tiene varias herramientas, las cuales se dividen en distintas "categorías". Analizaremos las herramientas chequeadoras de enlaces y de funcionalidad, dado que las otras categorías como ser: manejo de cookies, javascript, formularios, etc. [23], no aplican en la evaluación realizada en el presente trabajo.

4.1.4.1 Chequeadores de Enlaces

XENU [24]: Permite encontrar los enlaces rotos en un análisis en profundidad (HTTP y HTTPS). Como resultado muestra un listado de los enlaces (imágenes, .css, etc.) rotos en el sitio.
LINK Checker W3C [25]: Es una herramienta disponible On-Line. Permite encontrar enlaces rotos, anclas mal definidas, advertir sobre redirecciones, etc.
DRKSpider [26]: Permite la navegación por enlaces internos, externos; imágenes, .CSS y otros archivos. Genera como resultado un árbol jerárquico con los enlaces del sitio en prueba, con información detallada.
Link Evaluator [27]: Se integra a la interfaz del navegador. Se procesa sólo la página actual y no realiza un análisis en profundidad, resaltando en distintos colores el estado de los enlaces.

4.1.4.2 Funcionalidad

Selenium IDE [28]: Es una extensión para el navegador Web Firefox. Permite grabar clicks, tipeo y otras acciones para realizar test. Estos scripts grabados, luego pueden ser exportados en distintos lenguajes (PHP, JAVA, Ruby, C, etc.) para su posterior adaptación y utilización.
HTTPUnit [29]: Se basa en la metodología Extreme Programming. Se pueden realizar pruebas funcionales antes de que estén generadas las páginas Web. No se basa en los controles que tenga la página, si no que se basa en los valores de entrada que el usuario pueda ingresar.
Badboy [30]: Permite grabar y luego reproducir las acciones realizadas por los usuarios, luego este script puede ser utilizado en otras herramientas, como ser JMeter. Se puede integrar al navegador Web Internet Explorer.
SAHI [31]: Permite grabar y luego reproducir script. Tiene soporte funciones realizadas en lenguaje Javascript.

4.1.5 Herramientas para Pruebas de Rendimiento

Las herramientas, en este caso son útiles, por ejemplo, en las pruebas que necesitan tener concurrencia de usuarios al servidor a probar. Algunas de ellas son:
JMeter [32]: Es utilizada para realizar pruebas de rendimiento, de stress, de carga y de volumen, sobre recursos estáticos o dinámicos (Servlets, scripts Perl, Objetos Java, BB.DD., Servidores de FTP, etc.).
OpenSTA [33]: Se pueden crear script con peticiones HTTP y HTTPS, para realizar pruebas de rendimiento. Permite captar las peticiones del usuario generadas en un navegador Web, luego guardarlas, y poder editar para su posterior uso.
WEbLoad [34]: Permite realizar pruebas de rendimiento, a través de un entorno gráfico en el cual se pueden desarrollar, grabar y editar script de pruebas.
Grinder [35]: Es un framework escrito en Java, con el cual se pueden realizar pruebas de rendimiento, a través de script escritos en lenguaje Jython [36]. Permite grabar las peticiones del cliente sobre un navegador Web para ser luego reproducido.

5.2 Tablas Comparativas

Las herramientas antes mencionadas, fueron analizadas y las siguientes tablas resumen las características más destacadas de cada una de ellas. Esto permite contar con una guía que nos permitirá evaluar su utilización dependiendo de la finalidad en la cual se van a utilizar. La columna "Documentación" se refiere siempre a la publicada en los sitios oficiales o reconocidos.

Tabla 1. Herramientas para pruebas de aceptación

Herramienta	UI	Licencia	Plataforma	Lenguaje	Última actual.	Documen-tación
FitNesse	WEB	GPL	Windows / Linux	Java, C#, PHP, Ruby, .NET, etc.	Julio 2009	Guía de Usuario
Avignon	GUI	GPL	Windows /	Java, .NET, etc.	Octubre	Sin Datos

	Linux	2009

Tabla 2. Herramientas para pruebas estáticas de código

Herramienta	UI	Licencia	Plataforma	Lenguaje	Última actual.	Documen-tación
PHPLint	GUI	BSD[3]	Windows/ Linux	PHP	Mayo 2009	Tutorial / Manual
RATS	CLI	GPL	Windows/ Linux	C++, Perl, PHP y Python	Sept. 2009	Sin Datos
YASCA	CLI	GPL	Windows/ Linux	Java, .NET, PHP, HTML, CSS, etc.	Mayo 2009	Manual
PMD	CLI	BSD[3]	Windows/ Linux	Java	Febrero 2009	Tutorial/ Manual
FindBugs	GUI/ CLI	GPL	Windows/ Linux	Java	Marzo 2009	Tutorial/ Manual

Tabla 3. Herramientas para pruebas unitarias

Herramienta	UI	Licencia	Integración	Última actual.	Documentación
JUnit	Integrada	CPL	Eclipse NetBeans	Mayo 2009	Coockbook/Foro /FAQ
PHPUnit	CLI	PHP	No Aplica	Junio 2009	Manual
SimpleTest	CLI[4]	LGPL	Eclipse	Abril 2008	Tutorial/How to

Tabla 4. Herramientas para pruebas funcionales – Chequeadores de enlaces.

Herramienta	UI	Licencia	Procesa-miento	Plataforma	Última actual.	Documenta-ción
XENU	GUI	Freeware	Remoto/ Local	Windows	Abril 2009	FAQ
LINK Checker W3C	WEB	GPL	Remoto	Windows/Linux	No Aplica	Manual
DRKSpider	GUI	GPL	Remoto/ Local	Windows	Abril 2009	Foro
Link Evaluator	WEB	Apache License	Remoto/ Local	Windows/Linux	Mayo 2009	How to/Ejemplos

Tabla 5. Herramientas para pruebas funcionales – Funcionalidad.

Herramienta	UI	Licencia	Plataforma	Última actual.	Documenta-ción
Selenium	GUI	Apache	Varios[5]	Junio 2008	Tutorial/

[3] Equivalente a licencia BSD
[4] Puede integrarse también a la herramienta Eclipse
[5] Cualquier SO en el cual tenga instalado el navegador Web Firefox

IDE					Manual/Wiki
HTTPUnit	WEB	Propia	Windows/Linux	Mayo 2008	Tutorial/ Manual/FAQ
Badboy	WEB	LGPL	Windows	Diciembre 2008	Manual/ Foro
Sahi	GUI	Apache	Windows/Linux	Mayo 2009	Manual/ FAQ

Tabla 6. Herramientas para pruebas de rendimiento.

Herramienta	UI	Licencia	Plataforma	Concurrencia de usuarios	Última actual.	Documenta-ción
JMeter	GUI	Apache License	Windows / Linux	Si	Junio 2009	Tutorial
OpenSTA	GUI	GPL	Windows	Si	Octubre 2007	Guía de Usuario
WebLoader	GUI	GPL[6]	Windows	Si	Abril 2007	Tutorial
Grinder	GUI	GPL	Windows / Linux	Si	Febrero 2009	Guía de usuario/FAQ

6 Conclusión

En este trabajo se listaron y analizaron algunos criterios para la realización de pruebas en aplicaciones Web utilizando herramientas de código abierto. Estas herramientas no sólo se encuentran disponibles para su uso, sino que también se las podría adaptar a los requerimientos propios del proyecto.

La selección de las herramientas a utilizar dependerá del tipo de proyecto en el cual se las vayan a aplicar. Trabajar con las herramientas seleccionadas inicialmente puede ser una tarea muy compleja (por el tiempo requerido para su aprendizaje), pero principalmente teniendo en cuenta el tipo de aplicaciones analizadas en este trabajo, su uso permite minimizar los tiempos de testeo (dado que agilizan el proceso de realizar las repeticiones, por ejemplo), y aumentar la eficiencia y la calidad tanto del software como del equipo de desarrollo.

En los cuadros comparativos se reflejó tanto el análisis de las interfaces de usuario, especialmente si proveen o no interfaces gráficas y si proveen suficiente documentación.

Como puede verse a lo largo de este artículo, existen muchas herramientas que ayudan al equipo de testing de un proyecto. Elegir las más adecuadas no es una tarea sencilla. Este trabajo permite brindar un panorama general de las herramientas más populares y utilizadas para los distintos tipos de pruebas, haciendo especial hincapié en la importancia de su uso.

[6] Existen dos tipos de licencias: gpl y profesional

7 Referencias

1. The art of software testing, Glenford J. Myers - 2da Edición.
2. World Wide Web Consortium, http://www.w3c.org
3. L. Olsina, G. Lafuente, G. Rossi. "Specifying Quality Characteristics and Attributes for Websites." Lecture Notes in Computer Science 2016 Springer 2001, pag. 266 – 278.
4. http://www.informandote.com/jornadasIngWEB/articulos/jiw01.pdf
5. http://in2test.lsi.uniovi.es/repris/actividades/TestingJTS2007.pdf
6. Ingeniería del Software: Un Enfoque Práctico, Roger Pressman – 5ta Edición
7. http://www.owasp.org/index.php/OWASP_Testing_Guide_v2_Table_of_Contents
8. "What Is Software Testing? And Why Is It So Hard?" (IEEE)
9. http://www.istqb.org/download.htm
10. http://www.aptest.com/resources.html
11. http://www.opensourcetesting.org/
12. http://www.testingfaqs.org/
13. http://fitnesse.org/
14. http://www.nolacom.com/avignon/index.asp
15. http://www.icosaedro.it/phplint/
16. http://www.fortify.com/security-resources/rats.jsp
17. http://www.yasca.org/
18. http://pmd.sourceforge.net/
19. http://findbugs.sourceforge.net/
20. http://www.junit.org/
21. http://www.phpunit.de/
22. http://www.simpletest.org/
23. http://www.softwaretestinghelp.com/web-application-testing/
24. http://home.snafu.de/tilman/xenulink.html
25. http://validator.w3.org/checklink
26. http://www.drk.com.ar/index.php
27. https://addons.mozilla.org/es-ES/firefox/addon/4094
28. http://seleniumhq.org/projects/ide/
29. http://httpunit.sourceforge.net/index.html
30. http://www.badboy.com.au/
31. http://sahi.co.in/w/
32. http://jakarta.apache.org/jmeter/
33. http://www.opensta.org/
34. http://www.webload.org
35. http://grinder.sourceforge.net/
36. http://www.jython.org/
37.http://www.cs.colostate.edu/icst2008/AnnieKeynote.pdf