Indicadores de la Utilización del Bug Traker en Proyectos F/OSS

Jorge Ramirez[1], Gustavo Gil, Gonzalo Romero y Loraine Gimson

[1] Proyecto de Investigación 1690 - CIUNSa - Universidad Nacional de Salta, Argentina
{ramirezj, gdgil, dgromero,loraine}@cidia.unsa.edu.ar

Resumen. El presente trabajo explora las características de la información sobre detección y corrección de errores en proyectos de software libre y de código abierto (F/OSS) alojados en SourceForge, con el fin de detectar posibles indicadores de la confiabilidad de la información para la evaluación de proyectos de este tipo, con miras a la reutilización de código. Para ello, hemos recolectado la información del Bug-tracker de 15 proyectos de dominios similares, caracterizados por alta actividad de desarrollo y analizamos las limitaciones y posibilidades que ofrece la información accesible.

Palabras Clave: Software Libre y de Código Abierto, reutilización, ciclo de vida de errores, indicadores, herramientas de seguimiento de errores

1 Introducción

El software libre y de código abierto (F/OSS por sus siglas en inglés) constituye una fuente potencial de recursos para los desarrolladores, ya sea desde la perspectiva de la reutilización como componentes [1] o como base para adaptar o extender una aplicación. Ambos movimientos (Software Libre [2] y de Código abierto [3]) ubican a la reutilización como uno de los objetivos centrales, en virtud de la libre disposición del código fuente.

Sin embargo, la enorme heterogeneidad de los proyectos, estilos de desarrollo e información pública disponible, dificultan la adopción de estrategias que contemplen tal reutilización.

Entre la información habitualmente disponible en los proyectos a los que nos referimos, se destaca el seguimiento del reporte y corrección de fallas. Estos datos podrían constituir indicadores de la efectividad de remoción de fallas en un proyecto F/OSS.

Para el presente trabajo, exploramos la información disponible sobre seguimiento de fallas (bug tracking) en un conjunto de proyectos alojados en SourceForge, el mayor repositorio de código abierto.

El trabajo se organiza de la siguiente manera: en primer lugar se resume una serie de investigaciones relacionadas con el tema que nos ocupa. En el apartado 3

describimos las características del sistema de tracking que brinda SourceForge a los proyectos que se hospedan allí; a continuación detallamos la metodología que hemos seguido, exponemos la información que hemos relevado y presentamos la herramienta utilizada para la tarea. En la sección 4 analizamos la información obtenida. En la sección 5 presentamos las conclusiones y proponemos futuras líneas de investigación.

2 Panorama de la Investigación en el Tema

Eric Raymond enunció la llamada "Ley de Linus", según la cual "*Dado un número suficientemente elevado de ojos, todos los errores se convierten en obvios*". Esta hipótesis apunta a la confiabilidad de un producto de F/OSS en función de la libre participación de los interesados en el mecanismo que Raymond caracteriza como el "Bazar" (en oposición al modelo de desarrollo cerrado, al que asocia con la construcción de una catedral) [4].

Michlmayr y Senyard [5] destacaron que pese a la abundancia de información disponible, existen pocos trabajos que la estudien. En su trabajo, los autores mencionados realizaron un estudio estadístico profundo del proceso de informe y corrección de errores en el sistema Debian, desde 1994 hasta fines de 2003, observando alrededor de 200.000 reportes. Entre otros aspectos, Michlmayr y Senyard concluyen que los datos obtenidos sostienen la hipótesis de Raymond mencionada anteriormente.

Kim y Whitehead [6] consideran que el tiempo de resolución de un error es una medida poco utilizada en comparación con la cantidad de errores. Los autores analizan las estadísticas de resolución de errores en archivos de los proyectos ArgoUML y PostgresSQL durante un período de tiempo considerable (1 año en el primer caso, 4 años en el segundo). Para identificar las fallas, rastrearon las palabras claves "bug" y "fixed" en los archivos de registros de cambios (change logs) o a través de la referencia al informe de error correspondiente. Concluyen en que la resolución de la mitad de los errores requiere entre 100 y 300 días.

Asundi y Jayand [7] estudiaron los diferentes roles de los desarrolladores en el proceso de corrección de errores; para ello analizaron las listas de correo de cuatro proyectos de características diferentes en cuanto a tamaño, destinatarios, y número de desarrolladores; en el caso del proyecto Gaim (un popular cliente libre de mensajería en Internet), recopilaron manualmente la información de la interfaz Web del bug tracker del proyecto en SourceForge.

La recopilación de información pública en repositorios como SourceForge no está exenta de peligros y limitaciones, según advierten Howison y Crowston [8]. Estos autores analizaron 140 proyectos alojados en el repositorio mencionado. En dichos proyectos contaron 7 o más desarrolladores y 100 o más errores reportados en el tracking. Ellos exponen una serie de previsiones necesarias para la extracción de información mencionando, entre otras dificultades, aquellas causadas por datos truncos o erróneos, por la inconsistencia en la utilización de los recursos de desarrollo

en algunos proyectos y por las complicaciones surgidas de la migración de la información de un proyecto desde otros sistemas.

Las pretensiones de extender conclusiones basadas en esa información a otros ámbitos, particularmente a proyectos de código cerrado, también encuentra importantes restricciones, según revelan Yu, Schach y Chen [9] en un trabajo sobre la medición de la mantenibilidad en proyectos de código abierto. Esos autores advierten que las condiciones en que se producen los datos en un proyecto de código abierto no son asimilables a los procesos de un desarrollo cerrado. Respecto de la información obtenida del seguimiento de errores, cuestionan la falta de vinculación con versiones concretas del software, el desfasaje entre la antigüedad de los datos disponibles y la fecha en que se produjeron los primeros lanzamientos y la ausencia de elementos que permitan estimar el esfuerzo que requiere la solución de errores.

Diversos trabajos pusieron la mira en las características de los procesos de informe o detección y corrección de errores en el ámbito del F/OSS. Ahmed y Gokhale [10] estudiaron el ciclo de vida de los errores en el desarrollo del núcleo de Linux, con el fin de detectar las condiciones en que se producen y se resuelven. Hooweijer y Westley [11] analizaron más de 27.000 informes de error del sistema de tracking del proyecto Mozilla Firefox, advirtiendo las imprecisiones e inconsistencias que se encuentran en los informes de errores y las consecuencias que ello tiene en la corrección.

Estas dificultades no son exclusivas del ámbito F/OSS; Aranda y Venolia [12] observaron omisiones y datos incorrectos analizando repositorios de seguimientos de bugs en proyectos de Microsoft. Los autores sostienen en ese trabajo que tales deficiencias no son exclusivas de la empresa mencionada.

En la misma línea de este paper Lintula, Koponen y Hotti [13] recopilaron datos del Tracker de SourceForge para 4 proyectos F/OSS grandes (PhpMyAdmin, Gaim, Azureus y PDFCreator). El objetivo de esos autores era el de conocer los procesos de mantenimiento que revela esa información. A los fines de este trabajo, destacamos el análisis del ciclo de vida de un bug y las importantes diferencias que revelan en cuanto a la utilización de la herramienta de seguimiento de que disponen.

3 Recolección de Datos

Para el presente trabajo obtuvimos datos de los sistemas de seguimiento de errores de 15 proyectos diferentes. Antes de analizar los datos en cuestión, es necesario exponer las características del sistema de tracking que ofrece SourceForge. Posteriormente, explicaremos la forma en que recopilamos la información

3.1 El Sistema de Tracking de SourceForge

SourceForge ofrece a los administradores de proyectos un sistema de tracking que permite el seguimiento de cualquier tipo de asunto que requiera el manejo de

propiedad, estado y actividad [14]. El administrador puede crear diferentes trackers para administrar informes de errores, solicitudes de nuevas funcionalidades, etc.

Cada entrada de un tracker es un "artefacto". Los usuarios o desarrolladores que quieren comunicar un error, deben crear un artefacto, ingresando un resumen del problema y una descripción breve. Si el administrador ha definido grupos o categorías, podrá vincularse el informe con una versión o módulo específicos. El autor del reporte podrá también adjuntar un archivo con su correspondiente descripción (por ejemplo, una captura de pantalla).

El administrador puede reflejar los cambios que se produzcan en relación con el artefacto otorgando uno de los siguientes estados: pending, deleted o closed.

Los campos por defecto de cada registro de tracker son los siguientes: ID, Summary, Status, Opened, Assignee, Submitter, Resolution, Priority. El campo Resolution puede tomar los valores Fixed, Accepted, None, Invalid, Out of Date, Rejected, Won't Fix, Works for Me, Duplicate, Postponed, Later. Cualquier usuario (incluso un visitante) puede acceder a la información específica de cada artefacto, en una pantalla particular donde aparece la descripción completa, comentarios de usuarios y desarrolladores y una tabla de cambios, donde figuran los diferentes estados y variaciones por los que pasó el artefacto en cuestión.

3.2 Recopilación de los Datos

Para extraer las informaciones desde la interfaz Web, desarrollamos la aplicación sencilla Bufo (Bug Fixing in Open-Source) escrita en java. Dicha aplicación está disponible en http://sourceforge.net/projects/bufo.

La fecha de cierre de las fallas que figuran como corregidas ("fixed") consta en la página individual del artefacto; por ello, el programa obtiene de allí el dato mencionado. Ese dato resulta particularmente relevante desde la perspectiva de estimar la eficiencia en la remoción de fallas de un proyecto.

Los resultados de la exploración del bug tracker de cada proyecto se almacenan en un archivo CSV para facilitar el análisis posterior.

Los proyectos elegidos fueron los 15 mejor rankeados de la categoría Enterprise de SourceForge, el 29 de mayo de 2009. Pese a integrar la misma categoría, los proyectos en cuestión son muy heterogéneos en cuanto a dominio, cantidad de desarrolladores y actividad. La selección de los mejor rankeados obedece a la presunción de que tales proyectos contarían con informaciones más frecuentes, por lo que el registro de errores podría ser más relevante.

El ranking de los proyectos de SourceForge surge de ponderar un conjunto de variables referidos al tráfico (donde se toma en cuenta la cantidad de hits sobre el sitio del proyecto, sobre el logo, el número de descargas), el desarrollo (envíos de código al repositorio SVN correspondiente, ingresos del administrador, archivos subidos) y comunicación (informes registrados en los trackers, envíos a las listas de discusión y envíos a los foros del proyecto) [15]. El cómputo se realiza en base a la actividad de los últimos 7 días.

4 Datos Obtenidos

El resumen de los datos obtenidos es el que se presenta en la siguiente tabla.

Tabla 1. Datos obtenidos.

Proyecto	Fecha de registro	Total de bugs	Closed	Open	Fixed
OpenBravo		-	-	-	-
PostBooks ERP	14/05/07	7769	6461	1306	4693
Web-ERP	07/01/03	216	206	7	53
AD-Empiere	09/09/06	2021	1547	429	955
openSwing	17/05/06	111	105	3	63
Osmius	24/02/06	186	156	27	123
Zenoss	20/03/06	3693	1272	2396	0
Openi	01/07/05	251	209	35	170
OpenGoo	12/03/07	199	191	6	97
LiVES	08/10/02	71	51	20	32
Inforama	29/05/08	99	86	13	77
OpenXava	02/11/04	45	32	11	2
PdfSam	15/02/06	45	38	6	23
Pin'em up	22/01/07	3	3	0	2
OpenTaps ERP	10/08/05	368	328	37	164

Como se puede observar, la cantidad de información es extremadamente variable. El primer proyecto en la tabla, OpenBravo, utiliza un sistema de seguimiento propio, de modo que la información que consta en SourceForge está desactualizada. El segundo, PostBooks, migró la información desde un sistema Mantis al tracker de SourceForge, por lo que se advierte que algunos artefactos que figuran como resueltos no tienen descripción ni puede accederse a la pantalla particular del mismo.

Llama la atención también la falta de artefactos definidos como solucionados (fixed) en el proyecto Zenoss.

Como advierten Howison y Crowston en el trabajo mencionado más arriba, las migraciones de datos constituyen una fuente de inconsistencias en la información.

5 Análisis de los Datos

Los datos obtenidos se refieren a características externas de los proyectos, antes que a mediciones específicas del software producido por ellos.
Pese a que se trata de una muestra limitada, queda en evidencia la heterogeneidad de los datos obtenidos.

En algunos proyectos se advierte una tasa de resolución de bugs extremadamente baja, destacándose Zenoss con ningún bug consignado como resuelto (pese al alto número de reportes que alberga). En cambio, se observa un comportamiento bastante similar en varios proyectos.

Consideramos aquí la cantidad de bugs consignados como solucionados respecto de aquellos en estado abierto o cerrado; desestimamos en esta cuenta los que figuran como pendientes o eliminados, ya que reflejan que los desarrolladores no tienen prevista la solución de los mismos, al menos en el corto plazo. Si tomamos en cuenta al conjunto de 14 proyectos (descartando a OpenBravo), se ve que el porcentaje promedio de bugs solucionados es del 50,68%, con una desviación estándar de 19,12. Sin embargo, si nos limitamos a aquellos proyectos que reflejan en el tracker una actividad más constante (los que tienen más de 50 artefactos), el porcentaje es de 54,59% con una desviación estándar de 15,3.

Luego de una primera mirada a la información, y teniendo en cuenta las observaciones de Lintula, Koponen y Hotti, analizamos la cantidad de bugs que constan como cerrados pero que no tienen asignado ningún valor de resolución. El detalle de los datos en cuestión se consigna en Tabla 2.

Otra información que puede relativizar los datos obtenidos es la fecha en que se informó el último bug y la fecha más reciente en que se consignó un bug como corregido; en casos en los que la proporción closed/none es baja, podría ocurrir que no se registren reportes desde hace mucho tiempo, lo cual puede sugerir que el proyecto alcanzó una alta eficacia en la corrección de errores habida cuenta de que tratamos con proyectos que presentan importante actividad de desarrollo (según se desprende del ranking asignado por SourceForge). No obstante, cabe la posibilidad de que el seguimiento de errores de algunos proyectos hayan migrado a otro sistema o acaso los errores latentes afectan casos de uso poco frecuentes.

En la lista relevada se destaca la situación de Pin'em up, con sólo 3 reportes de error en el tracker, el último de los cuales data del año 2007; sin embargo, la última versión lanzada fue publicada el 11 de mayo de este año

Tabla 2. Porcentaje de bugs cerrados sin resolución

Proyecto	closed/none	Fecha último bug reportado	Fecha último bug corregido
Zenoss	34,44%	29/05/09	-
WinRun4J	0,00%	12/03/09	24/02/08
OpenXava	64,44%	28/05/09	28/08/08
Pin'em up	33,33%	14/09/07	31/01/07
PdfSam	2,22%	21/05/09	28/03/09
LiVES	42,25%	01/06/09	05/06/09
Web-ERP	59,72%	14/05/09	23/03/09
openSwing	28,83%	23/05/09	19/05/09
Inforama	2,02%	22/05/09	13/05/09
OpenGoo	35,18%	25/11/08	14/04/09
Osmius	8,06%	18/05/09	06/05/09
OpenTaps ERP	19,84%	14/06/09	12/05/09
Openi	5,58%	15/05/09	20/05/09
AD-Empiere	18,81%	09/06/09	09/09/09
PostBook ERP	22,91%	15/06/09	13/06/09

Aquí debemos considerar que al configurar un tracker el administrador establece el tiempo durante el cual un artefacto permanecerá abierto en caso de que no haya actividad sobre él. Esto significa que un artefacto puede no haber sido cerrado por los desarrolladores, sino que podría haberse cerrado porque expiró su plazo.

Puede haber diferentes causas por las cuales el bug tracker de un proyecto tenga un porcentaje alto de artefactos cerrados sin resolución explícita; esos valores pueden ocasionarse porque el administrador o los desarrolladores no registran regularmente el estado final de los bugs cerrados, o porque la tasa de resolución de defectos es baja. En cualquier caso, un valor muy bajo desacredita la validez de la información del tracker respecto de la capacidad de un proyecto F/OSS de resolver los defectos que se detectan.

6 Conclusiones y trabajos futuros

Este trabajo constituye una primera aproximación a la recopilación de información públicamente accesible en proyectos F/OSS, con miras a brindar elementos de juicio

sobre la confiabilidad de dichos proyectos. Específicamente, hemos puesto la mira en información atinente a la mantenibilidad.

Lo primero que podemos concluir es que los datos que se recojan sobre un proyecto F/OSS en particular debe examinarse cuidadosamente, habida cuenta de la heterogeneidad no sólo de las características generales de los proyectos sino respecto de la utilización que hacen los diferentes equipos de desarrollo de las herramientas de gestión provistas por el repositorio (en este caso, SourceForge).

El porcentaje de bugs cerrados sin resolución constituye un indicativo sencillo de la gestión del seguimiento de errores de un proyecto concreto. Un valor alto no permite afirmar que exista una gestión deficiente, pero sí advierte sobre la posible inconsistencia de la información que ofrece el sistema de seguimiento. Ese valor puede deberse a la utilización inconsistente del tracker (por ejemplo, porque los desarrolladores no modifican el valor de la resolución), a que el seguimiento de errores se realiza a través de otros medios o a la baja efectividad en la corrección de fallas.

Los datos relevados sugieren cierta estabilidad en el desempeño respecto de la corrección de errores en proyectos muy diferentes, en tanto se observe continuidad en el seguimiento de los artefactos. En próximos trabajos podría analizarse ese desempeño en una cantidad mayor de proyectos, limitados a aquéllos con baja proporción closed/none, un número importante de reportes de error y actividad reciente de gestión de bugs.

La información recabada tiene carácter claramente preliminar. A partir de los mismos datos generados por la herramienta utilizada, sería posible analizar las variaciones en el tiempo promedio de resolución de los errores y estudiar la incidencia de la prioridad asignada a cada artefacto con relación al indicador mencionado. También sería interesante observar si el porcentaje de closed-none varía a lo largo del tiempo, y estudiar las vinculaciones de este indicador con el lanzamiento de versiones y con diferentes métricas generales de los proyectos F/OSS (por ejemplo, el número de desarrolladores, la frecuencia del lanzamiento de versiones).

Otro campo que podría ser fértil sería la búsqueda de correlaciones entre los datos referidos a la corrección de errores y los valores de diversas métricas de complejidad y/o mantenibilidad aplicables a los productos de software analizados. Las advertencias de Yu, Schach y Chen no tuvieron en cuenta que existen proyectos que vinculan los artefactos del tracker con versiones específicas e incluso con módulos o funciones particulares; no obstante, la utilización consistente del sistema de seguimiento es responsabilidad de los administradores de proyecto, y está librado a la variedad característica del desarrollo de F/OSS.

Estas futuras investigaciones pueden orientar a los administradores de proyectos F/OSS respecto de la planificación de la utilización del tracker y ofrecer un conjunto de información empírica que, tratada con el rigor necesario, puede ser de interés para diversos campos de la ingeniería de software. En cuanto a la evaluación de proyectos F/OSS, podrán aportar elementos sobre la eficacia en la corrección de errores para un proyecto determinado, constituyendo a la vez una referencia respecto de la modificabilidad de los productos lanzados por esos emprendimientos.

Referencias

1. Hissam S.: Weinstock CB;Plakosh D & Asundi J:Perspectives on open source software. (2001)
2. Free Software Foundation: Free software definition. http://www.fsf.org/licensing/essays/ /freesw.html.
3. Open Sorce Initiative: http://www.opensource.org/docs/osd.
4. Raymond E.: The cathedral \& the bazaar. http://catb.org/~esr/writings/cathedralbazaar/ cathedralbazaar/.
5. Michlmayr M. & Senyard A.: A statistical analysis of defects in {d}ebian and strategies for improving quality in free software projects. In The economics of open source software development. (2006).
6. Kim S. & Whitehead Jr. E.: How long did it take to fix bugs?. In Msr '06: proceedings of the 2006 international workshop on mining software repositories. 2006.
7. Asundi J. & Jayant R.: Patch review processes in open source software development communities: a comparative case study. Hawaii International Conference on System Sciences .0: p. 166c (2007)
8. Howison J. & Crowston K.: The perils and pitfalls of mining {sourceforge}. In {proc. Of mining software repositories workshop at the international conference on software engineering (icse)}.(2004).
9. Yu Li;Schach S & Chen K.: Measuring the maintainability of opensource software. In Empirical software engineering, 2005. 2005 international symposium on. 2005.
10. Ahmed M & Gokhale S.: Linux bugs: life cycle and resolution analysis. In Qsic08. 2008.
11. Hooimeijer P. & Weimer W.: Modeling bug report quality. In Ase '07: proceedings of the twentysecond ieee/acm international conference on automated software engineering. 2007.
12. Aranda J. & Venolia G.: The secret life of bugs: going past the errors and omissions in software repositories. In Icse '09: proceedings of the 2009 ieee 31st international conference on software engineering. (2009).
13. Lintula H., Koponen T. & Hotti V.: Exploring the maintenance process through the defect management in the open source projects four case studies. In Icsea '06: proceedings of the international conference on software engineering advances. (2006).
14. Sourceforge enterprise edition user guide. (2006).
15. SourceForge.net: Message sent to all sourceforge.net project administrators, 2005/05/05. http://sourceforge.net/forum/forum.php?forum_id=465092.