Diferencias entre las Actividades de Mantenimiento en los Procesos de Desarrollo Tradicional y Open Source

John W. Castro[1], Silvia T. Acuña[1], Oscar Dieste[2]

[1]Departamento de Ingeniería Informática, Universidad Autónoma de Madrid
Calle Francisco Tomás y Valiente 11, 28049 Madrid, España
john.castro@estudiante.uam.es, silvia.acunna@uam.es
[2]Facultad de Informática, Universidad Politécnica de Madrid
Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, España
odieste@fi.upm.es

Resumen. ***Antecedentes***. La creciente importancia del *Open Source Software* (OSS) ha llevado a los investigadores a estudiar cómo los procesos OSS difieren de los procesos de la ingeniería del software tradicional. ***Objetivo***. Determinar las diferencias y similitudes entre las actividades del proceso de mantenimiento seguido por la comunidad OSS y el establecido por el estándar IEEE 1074:2006. ***Método***. Para conocer las actividades que conforman el proceso de desarrollo OSS realizamos un *Systematic Mapping Study*. Posteriormente, realizamos un emparejamiento entre las actividades del estándar IEEE 1074:2006 con las actividades del proceso OSS. ***Resultados***. Encontramos un total de 22 estudios primarios. De estos estudios, el 73% contaban con actividades relacionadas con el proceso de mantenimiento. ***Conclusiones***. El proceso de mantenimiento tradicional del software no encaja con lo que ocurre en la comunidad OSS. En su lugar, puede ser mejor caracterizar la dinámica general de la evolución OSS como reinvención. Esta reinvención emerge continuamente de la adaptación, aprendizaje, y mejora de las funcionalidades y calidad del OSS. Los proyectos OSS evolucionan a través de mejoras menores donde participan tanto usuarios como desarrolladores.

Palabras clave. *Systematic Mapping Study*, Proceso de Mantenimiento, *Open Source Software*.

1 Introducción

La creciente importancia del *Open Source Software* (OSS) en los últimos años ha llevado a los investigadores a estudiar cómo los procesos OSS difieren de los procesos de ingeniería del software tradicional. La comprensión del contexto, estructura y actividades de los procesos de desarrollo OSS que se encuentran en la práctica ha sido y sigue siendo un problema difícil de resolver [15].

Algunos estudios han demostrado que los procesos OSS son diferentes en muchos aspectos de los procesos software tradicionales [15][16][11]. Por otro lado, algunos autores como Fuggetta [4] y Godfrey y Tu [5] no son de tal opinión y afirman que el

modelo de desarrollo OSS no es exactamente una nueva descripción del proceso, sino sólo una visión alternativa de las actividades de ingeniería del software aplicadas a los modelos de desarrollo comercial tradicional. En cualquier caso, podemos fácilmente encontrar numerosos ejemplos de sistemas OSS exitosos, como los estudiados por Mockus et al. [8][9]. Por tanto, los procesos seguidos por la comunidad OSS objetivamente funcionan ya que obtienen resultados de calidad reconocidos a nivel mundial, lo cual a su vez implica que una mejor comprensión de los mismos puede informar y favorecer los procesos de desarrollo de software más habituales.

Existen pocos trabajos que analicen el proceso de desarrollo OSS. El trabajo de [3] realiza una revisión de la literatura relacionada con el proceso de desarrollo OSS, en particular sobre las prácticas seguidas por los equipos de desarrollo OSS durante el análisis de requisitos, la implementación, las pruebas, la liberación de versiones y el mantenimiento. El proceso de requisitos seguido en proyectos de desarrollo OSS en diferentes dominios (como la astrofísica, juegos de ordenador en red y sistemas de diseño de software) ha sido estudiado por Scacchi y colegas [12][13][14]. En estos estudios se ha encontrado, por ejemplo, que por lo general no hay documentos de requisitos de software explícitos. Los procesos de requisitos, diseño e implementación seguidos por la comunidad OSS también han sido estudiados por Castro y Acuña [2]. Estos autores observan, por ejemplo, que hay ciertos factores comunes en todos los proyectos OSS: (1) la modularidad con la que se diseñan e implementan los sistemas software; (2) la implementación como prioridad en la comunidad OSS; y, (3) los desarrolladores eligen lo que desean diseñar según sus preferencias.

Realmente, ¿puede afirmarse que los procesos OSS difieren del modelo tradicional? Este trabajo apunta en la dirección de responder a esta inquietud, centrándonos en el proceso de mantenimiento, ya que los esfuerzos de la comunidad OSS apuntan principalmente en esa dirección [1]. Además, este trabajo es el segundo de una serie de trabajos donde estudiamos el proceso de desarrollo OSS. Esta serie de trabajos está basada en un único SMS. En el primero de estos [1], abordamos el estudio de los procesos de requisitos, diseño e implementación seguidos por la comunidad OSS. Nuestro objetivo es determinar las diferencias y similitudes existentes entre el proceso de mantenimiento seguido por la comunidad OSS y el establecido por el estándar IEEE 1074:2006 [6]. Este estándar es nuestro modelo de proceso de referencia, pues su influencia se extiende a muchos procesos de desarrollo y es lo que consideramos en este trabajo de investigación como desarrollo tradicional.

Para lograr este objetivo, primeramente necesitamos conocer las actividades que conforman el proceso de mantenimiento seguido por la comunidad OSS, para lo cual hemos realizado un *Systematic Mapping Study* (SMS), siguiendo el procedimiento propuesto por Kitchenham y colegas [7]. Un SMS es una metodología que consiste en investigar la literatura sobre un área de interés particular, con el objetivo de determinar la naturaleza, el alcance y la cantidad de estudios primarios publicados [10]. Hemos elegido esta metodología, porque nos permite identificar y categorizar la literatura mostrando una visión sintetizada del área de investigación que está siendo considerada, con el fin de dirigir la investigación futura. Posteriormente, hemos comparado estas actividades con las establecidas por el estándar IEEE 1074 [6]. Finalmente, he-

mos reconciliado las diferencias existentes y obtenido las características principales que caracterizan al proceso de mantenimiento OSS.

Este trabajo se organiza de la siguiente manera. En la sección 2 se describe el método de investigación. La sección 3 describe el emparejamiento realizado entre las actividades del proceso de mantenimiento del Estándar IEEE 1074 con las actividades análogas del proceso seguido por la comunidad OSS. En la sección 4 se realiza una comparativa entre estas actividades. La sección 5 presenta la discusión del proceso de mantenimiento seguido por la comunidad OSS. En la sección 6 se describen las amenazas a la validez de esta investigación, y finalmente en la sección 7 se presentan las conclusiones.

2 Método de Investigación

El método de investigación aplicado fue el SMS [10]. Este SMS tenía por objetivo responder a la siguiente pregunta de investigación:

RQ: ¿Qué actividades conforman los modelos de proceso OSS?

El proceso de SMS se inició con la identificación de las palabras clave y las cadenas de búsqueda las cuales se derivaron a partir de la pregunta de investigación planteada. Se realizó una búsqueda preliminar, a partir de la cual se obtuvieron algunos artículos que fueron estudiados para determinar los términos más apropiados para el SMS. Estos términos fueron validados y completados por dos expertos investigadores en el área de ingeniería del software. Las cadenas de búsqueda finalmente empleadas fueron:

- *Open Source* AND *Software Process Model*
- *Open Source* AND *Software Development Process*
- *Open Source* AND *Development Process*
- *Free Source* AND *Software Process Model*
- *Free Source* AND *Software Development Process*
- *Free Source* AND *Development Process*

Las bases de datos (BBDD) electrónicas usadas en el SMS fueron: *IEEE Xplore, ACM Digital Library, SpringerLink, Science Direct* y *Scopus*. Cada una de las seis cadenas de búsqueda definidas fue aplicada a cada una de las BBDD seleccionadas. La Tabla 1 presenta para cada una de estas BBDD electrónicas los campos utilizados en la búsqueda. Los campos utilizados no fueron siempre los mismos, ya que dependían de las opciones que brindaba cada una de las BBDD. Los campos *"Abstract"* y *"Title"* fueron utilizados en todas las BBDD.

Durante la búsqueda se fijó como fecha límite de publicación de los artículos el 31 de marzo de 2010. Para la determinación de los estudios primarios que son relevantes a nuestra pregunta de investigación, se utilizaron los siguientes criterios de inclusión y exclusión que se muestran en la Tabla 2.

Tabla 1. Campos de Búsqueda Empleados en cada BBDD

BBDD	Sitio Web	Campos de Búsqueda Seleccionados
IEEE Xplore	http://ieeexplore.ieee.org/	"Index Terms"
ACM	http://portal.acm.org/	"Abstract"
SpringerLink	http://www.springerlink.com/	"All Text OR Abstract"
Science Direct	http://www.sciencedirect.com/	"Abstract OR Title OR Keywords"
Scopus	http://www.scopus.com/	"Article Title OR Abstract OR Keywords"

Tabla 2. Criterios de Inclusión y Exclusión

Criterios de Inclusión
(El título del artículo debe contener las palabras '*open source*' o '*free source*' **OR** Las palabras clave hacían alusión al proceso de desarrollo OSS **OR** El resumen mencionaba alguna cuestión sobre el proceso de desarrollo OSS) **AND** (El artículo describe el proceso de desarrollo OSS **OR** El artículo presenta las actividades del proceso de desarrollo OSS **OR** El artículo presenta las actividades del proceso de desarrollo de software *free source* **OR** El artículo describe el proceso de desarrollo seguido en un proyecto particular OSS **OR** El artículo presenta una propuesta de un proceso de desarrollo OSS)
Criterios de Exclusión
(El artículo no describe el proceso de desarrollo OSS **OR** El artículo no presenta las actividades del proceso de desarrollo OSS **OR** El artículo no presenta las actividades del proceso de desarrollo de software *free source*)

La Tabla 3 muestra para cada BBDD el número de artículos obtenidos al aplicar las seis cadenas de búsqueda, así como el número de artículos candidatos seleccionados. Los artículos candidatos son todos aquellos estudios que cumplen con los criterios de inclusión pero aplicados únicamente sobre el título y las palabras clave. Esta estrategia nos ha permitido filtrar rápidamente el resultado de las búsquedas, al reducir de 12.267 a 619 el número de artículos que evaluar en detalle (solo un 5% del total). Este conjunto de candidatos no contiene duplicados.

Tabla 3. Número Total de Artículos Obtenidos en cada BBDD

Términos de Búsqueda	Encontrados	Candidatos	Estudios Primarios
IEEE Xplore	387	89	7
ACM Digital Library	6.118	282	8
SpringerLink	2.459	147	3
Science Direct	199	21	0
Scopus	3.104	80	1
TOTAL	**12.267**	**619**	**19**

A continuación, se obtuvieron 19 estudios primarios después de aplicar los criterios de inclusión y exclusión de forma rigurosa. Posteriormente, se revisaron las referencias de estos estudios primarios y se encontraron 8 nuevos estudios que no se encontraban en ninguna de las cinco BBDD utilizadas. Así, el número total de estudios primarios ascendió a 27. De estos 27 estudios primarios, se eliminaron 5 debido a que los autores no definían las actividades del proceso OSS, lo que impedía que no se pudiera realizar el análisis de las mismas. Finalmente, el número de estudios primarios utilizados fue de 22, los cuales se muestran en el Apéndice A. Se ha asignado un código a cada uno de dichos estudios primarios para facilitar su referencia en el presente trabajo.

3 Emparejamiento entre las Actividades del Estándar IEEE 1074 con las Actividades del Proceso OSS

De las actividades que conforman los procesos OSS nos hemos centrado en el análisis de las actividades relacionadas con el proceso de mantenimiento. Por una parte, el total de estudios primarios, más de la mitad (55%) estudian proyectos OSS particulares. De este 55% de estudios, más del 80% (10 de 12) cuenta con actividades relacionadas con el mantenimiento. Por otra parte, el 73% del total de estudios primarios encontrados (16 de 22), cuenta con actividades relacionadas con el mantenimiento. La revisión de dichos estudios nos permitió obtener el nombre (cuando estaba disponible) y la descripción de la actividad dada por cada autor. Según la descripción de cada actividad, realizamos un emparejamiento con la actividad análoga del estándar IEEE 1074 [6], teniendo en cuenta para ello no el nombre, sino el objetivo de la actividad de proceso OSS.

Al terminar el emparejamiento, obtuvimos una tabla donde cada una de las actividades del proceso OSS relacionadas con el proceso de mantenimiento fueron emparejadas con una actividad del estándar IEEE 1074. A esta tabla la denominamos *Tabla de Instanciación,* y contiene para cada actividad el nombre y la descripción dado por el autor, así como un comentario. La Tabla 4 ilustra solo un fragmento de esta tabla de instanciación, que por restricciones de espacio no podemos presentar de manera completa.

El emparejamiento supuso un gran esfuerzo, pues algunos autores no dividían la descripción del proceso de desarrollo OSS en actividades, sino que la presentaban como una narrativa, la cual tuvo que ser separada en fragmentos para facilitar su análisis y posterior emparejamiento. En otros casos, cuando los autores si dividían el proceso en actividades, los nombres de éstas no eran los apropiados para las definiciones que los mismos autores aportaban. Por esta razón, se decidió realizar el emparejamiento basándose únicamente en la descripción pero no en el nombre dado por el autor. Adicionalmente, algunas definiciones enmarcadas bajo un solo nombre de actividad debían ser procesadas por partes ya que el autor estaba definiendo realmente varias actividades. De estas dificultades encontradas se han definido las siguientes tipologías:

- CN-CC: Correspondencia **C**orrecta en **N**ombre, **C**orrecta en **C**ontenido.
- IN-CC: Correspondencia **I**nadecuada en **N**ombre, **C**orrecta en **C**ontenido.
- SN-CC: **S**in **N**ombre, **C**orrecta en **C**ontenido.

Tabla 4. Fragmento de la Tabla de Instanciación

Nombre Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Definición del Autor	Comentario
Implement Problem Reporting Method	1. Communication and documentation [Mart07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	También está en: - Perform Architectural Design - Report Evaluation Results - Reapply SPLCP
	⋮	⋮	⋮
⋮	⋮	⋮	⋮
Reapply SPLCP	1. Communication and documentation [Mart07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	También está en: - Perform Architectural Design - Report Evaluation Results - Implement Problem Reporting Method
	⋮	⋮	⋮
⋮	⋮	⋮	⋮

Partiendo de la Tabla de Instanciación anterior, se procedió a realizar un análisis para determinar si el nombre y la descripción dada por cada uno de los autores a las actividades fue el adecuado. Como resultado de este proceso obtenemos una nueva tabla que denominamos Tabla de Análisis, de la cual se muestra un fragmento en la Tabla 5. Hemos usado las mismas actividades que en la Tabla 4 para ilustrar cómo evoluciona la información en cada una de las tablas y el proceso seguido. Como se puede apreciar, algunos de los nombres de las actividades dados por los autores han sido resaltados en gris, pues el nombre no es el adecuado según la finalidad de la actividad y el emparejamiento realizado. También fragmentos de la definición de ciertas actividades han sido resaltados en gris porque no aportan nada a la definición de la actividad o porque sencillamente definen otra actividad; es decir, el autor ha mezclado actividades con un mismo nombre. Hemos empleado texto resaltado en gris porque en blanco y negro no es posible utilizar códigos de colores.

A continuación, la Tabla 6 presenta el emparejamiento de todas las actividades identificadas del proceso de mantenimiento OSS con las actividades del proceso de mantenimiento del estándar IEEE 1074. Cada una de estas actividades tiene asignada su tipología correspondiente según lo explicado anteriormente, y el código asignado al estudio primario donde se encuentra esta actividad. Por restricciones de espacio no presentamos en la Tabla 6 las definiciones dadas por cada autor a cada actividad.

Tabla 5. Fragmento de la Tabla de Análisis

Nombre Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Definición del Autor	Tipología
Implement Problem Reporting Method	1. Communication and documentation [Mari07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	IN-CC (Se corresponde con más de una actividad)
	⋮	⋮	⋮
⋮	⋮	⋮	⋮
Reapply SPI.CP	1. Communication and documentation [Mari07]	Another powerful tool in the software process is the bug or issue tracker. An issue tracker lets developers and users report software issues and provides a place to store feature requests so they're not forgotten. We chose the Web-based phpBugTracker system because it's easy to install, has a clean user interface, and can easily handle multiple projects. As developers and users create issues in the system, the issues are given a unique ID and can be assigned a severity and priority. The issue tracker's administrator can give developers extra privileges, so that they can assign status and resolutions to issues. The system also lets you attach files to issues, providing a way to give reproducible test cases and even source patches.	IN-CC (Se corresponde con más de una actividad)
	⋮	⋮	⋮
⋮	⋮	⋮	⋮

Tabla 6. Emparejamiento entre las Actividades del Estándar IEEE 1074 y las Actividades del Proceso de Mantenimiento OSS

Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Tipología	Código
Identify Software Improvement Needs	Opening up requirements	IN-CC	[Seny04]
	Consult newsgroups	IN-CC	[Mock02]
	Quality	IN-CC	[Erdo09]
	Identification of work to be done	CN-CC	[Dinh05]
	Phase III: Establishing open source development project	IN-CC	[Gurb06]
Implement Problem Reporting Method	Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging	IN-CC	[Raym99]
	Discovering that a problem exists	SN-CC	[Mock00]
	Parallel debugging	IN-CC	[Jorg01]
	Users regarding feature requests, bugs	CN-CC	[Wynn04]
	Change requests	CN-CC	[Mock02]
	Reporting problems or requesting enhancements	CN-CC	[Mock02]
	Consult newsgroups	IN-CC	[Mock02]
	Identifying work to be done	IN-CC	[Mock02]
	Tiered participation	IN-CC	[Simm03]
	Volunteer	IN-CC	[John01]

Tabla 6 (Continuación). Emparejamiento entre las Actividades del Estándar IEEE 1074 y las Actividades del Proceso de Mantenimiento OSS

Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Tipología	Código
Implement Problem Reporting Method (Continuación)	Maintenance as evolutionary redevelopment, reinvention and revitalization	CN-CC	[Scac04]
	Quality	IN-CC	[Erdo09]
	Communication and documentation	IN-CC	[Mart07]
	Identification of work to be done	IN-CC	[Dinh05]
	User-initiated change request	CN-CC	[Gurb06]
	Production of packages	IN-CC	[Mong04]
	Bug report	SN-CC	[Yama00]
	Fixing bugs or implementing the proposals	CN-CC	[Yama00]
	Fixing bugs	CN-CC	[Egan04]
	Implement new features	IN-CC	[Egan04]
Reapply SPLCP	Discovering that a problem exists	SN-CC	[Mock00]
	Reporting problems or requesting enhancements	CN-CC	[Mock02]
	Communication and documentation	IN-CC	[Mart07]

La Tabla 7 presenta un resumen de la tipología de las actividades OSS relacionadas con cada una de las actividades del proceso de mantenimiento del estándar IEEE 1074. Este resumen incluye por cada actividad del proceso de mantenimiento tradicional y por cada tipología, el número total de actividades OSS y el número total de estudios primarios diferentes con respecto a esta tipología.

Tabla 7. Resumen de la Frecuencia de las Actividades OSS por Tipología

Nombre de la Actividad en IEEE	Tipología	Nro. Total de Actividades OSS	Nro. Total de Estudios Primarios Diferentes
Identify Software Improvement Needs	CN-CC	1	1
	IN-CC	4	4
Implement Problem Reporting Method	CN-CC	7	6
	IN-CC	11	10
	SN-CC	2	2
Reapply SPLCP	CN-CC	1	1
	IN-CC	1	1
	SN-CC	1	1

Para la actividad *Identify Software Improvement Needs*, el 80% (4 de 5) del total de estudios primarios emparejados en esta actividad nombra erróneamente las actividades. En cuanto a la actividad *Implement Problem Reporting Method*: el 56% (10 de 18) del total de estudios primarios emparejados en esta actividad nombra erróneamente las actividades, mientras que el 33% (6 de 18) cuenta con nombres adecuados se-

gún la descripción dada por el autor. Finalmente, para la actividad *Reapply SPLCP* existe un estudio primario para cada una de las tipologías: CN-CC, IN-CC y SN-CC.

Finalmente, hemos comparado la definición dada por el estándar IEEE 1074 con la definición dada por cada autor según el emparejamiento realizado. De esta comparativa hemos obtenido las diferencias y similitudes entre los procesos de mantenimiento OSS y tradicional, las cuales se describen en el apartado siguiente.

4 Actividades Proceso de Mantenimiento del Estándar IEEE 1074 vs. Actividades Proceso de Mantenimiento OSS

La Tabla 8 ilustra un fragmento de las diferencias y similitudes encontradas entre las actividades del mantenimiento, que por razones de espacio no podemos presentar de manera completa.

Tabla 8. Fragmento de la Comparativa del Estándar IEEE 1074 y las Actividades del Proceso de Mantenimiento OSS

Nombre Activ. Estándar IEEE 1074 [6]	Nombre Actividad para el Autor	Actividad Proceso Tradicional y Definición Dada por el Autor	
		Diferencias	Similitudes
Implement Problem Reporting Method	1. Communication and documentation [Mart07]	En la comunidad OSS los reportes de problemas o solicitudes de mejoras pueden ser realizados por cualquier persona, incluidos los usuarios. Estos diagnostican problemas y a la vez sugieren correcciones, en forma de ideas o de código fuente. En el desarrollo tradicional los usuarios solo reportan los errores, no los corrigen.	El objetivo es el mismo: reportar los problemas encontrados.
	⋮	⋮	⋮
⋮	⋮	⋮	⋮
Reapply SPLCP	1. Communication and documentation [Mart07]	En la comunidad OSS los desarrolladores o los usuarios pueden modificar el estado de un problema (si esta resuelto o no). En el desarrollo tradicional esto no es así.	Se realiza un seguimiento de los problemas reportados. Existe un responsable de esta actividad de seguimiento.
	⋮	⋮	⋮
⋮	⋮	⋮	⋮

En la comunidad OSS cualquier persona, en cualquier momento, puede sugerir o aportar mejoras al sistema software [Gurb06][Mock02][Seny04]. Estas mejoras son comunicadas generalmente a través de listas de correos. Los usuarios finales OSS que actúan como desarrolladores o como los encargados de realizar el mantenimiento, producen continuamente estas mejoras. Las modificaciones o actualizaciones de los sistemas OSS, se convierten luego de revisiones en versiones que son recombinadas con otras para liberar una versión estable. Estas mejoras articulan y adaptan un sistema OSS a las necesidades de los usuarios-desarrolladores, mientras lo reinventan [Scac04]. En algunos proyectos OSS, son los miembros del equipo núcleo los que deciden si adicionan o no una nueva funcionalidad [Dinh05]. En el desarrollo tradicional, la identificación de estas mejoras se realiza con base en diferentes documentos, resultado de otras actividades relacionadas con la planificación del proyecto. Además, son identificadas las herramientas, técnicas y métodos para la aplicación de estas recomendaciones. En algunos proyectos OSS, gracias a sistemas de puntuación, los usuarios saben lo que obtendrán de cada mejora del software. La comunidad OSS

evalúa cada funcionalidad continuamente, y las puntuaciones cambian con el tiempo. De esta manera la calidad es visible [Erdo09]. El estándar IEEE 1074 establece que las recomendaciones de mejora incluyan su impacto en la calidad del software.

En la comunidad OSS, los reportes de problemas o solicitudes de mejoras pueden ser realizados por cualquier persona, incluidos los usuarios, quienes diagnostican problemas y a la vez sugieren correcciones, en forma de ideas o de código fuente [Egan04][John01][Mock02][Raym99][Simm03][Yama00]. Estos reportes de problemas o solicitudes de mejoras se realizan a través de diferentes medios, como listas de correo y grupos de noticias [Mock00]. No hay un repositorio central. Las solicitudes que se encuentran en las listas de correo tienen la mayor prioridad [Mock02]. En el desarrollo tradicional, los usuarios solo reportan errores, no los corrigen. Estos reportes tienen una estructura definida. No todas las personas involucradas en el desarrollo tienen acceso al reporte de mejoras. Los desarrolladores o los usuarios en la comunidad OSS pueden categorizar los problemas y las solicitudes de mejoras, así como también modificar el estado de un problema o un error [Mart07]. En el desarrollo tradicional los usuarios no realizan esta labor.

5 Discusión del Proceso de Mantenimiento OSS

El mantenimiento de software es un proceso generalizado y recurrente en las comunidades de desarrollo OSS como lo es en el desarrollo tradicional. Tal vez esto no es de extrañar teniendo en cuenta que el mantenimiento generalmente es visto como la principal actividad asociada a un sistema software en su ciclo de vida. Sin embargo, el proceso de mantenimiento tradicional del software no encaja con lo que ocurre en la comunidad OSS. Las diferencias encontradas entre el proceso de mantenimiento de software en el desarrollo tradicional y el desarrollo OSS son debidas a la propia naturaleza de cada uno de los procesos. En su lugar, puede ser mejor caracterizar la dinámica general de la evolución OSS como reinvención. La reinvención se produce al intercambiar, examinar, modificar y redistribuir los conceptos y técnicas que han aparecido en el desarrollo tradicional, en publicaciones científicas y libros de texto, conferencias y experiencias de los usuarios-desarrolladores al participar en múltiples proyectos OSS. De este modo, la reinvención es una fuente que emerge continuamente de la adaptación, aprendizaje, y mejora de las funcionalidades y calidad del OSS [Scac04]. No hay distinción entre mantenimiento correctivo y evolutivo.

Los sistemas OSS evolucionan a través de mejoras menores o transformaciones que son expresadas, recombinadas, y redistribuidas a través de múltiples liberaciones con ciclos de vida cortos. Los usuarios finales OSS que actúan como desarrolladores (o como los encargados de realizar el mantenimiento) producen continuamente estas transformaciones. Cualquier persona dentro de la comunidad OSS puede reportar errores, o solicitar mejoras al sistema software. Algunos proyectos OSS como Apache cuentan con sus propias herramientas para el reporte de errores o problemas [Mock02]. Las modificaciones o actualizaciones se expresan como versiones *alpha*, beta que son redistribuidas y revisadas. Estas versiones pueden ser recombinadas con otras transformaciones para liberar una nueva versión estable. Las transformaciones

articulan y adaptan un sistema OSS a lo que sus usuarios-desarrolladores desean que haga, mientras reinventan el sistema [Scac04]. En algunos proyectos OSS, uno o dos desarrolladores realizan el seguimiento periódico de las nuevas peticiones, la eliminación de los reportes de problemas equivocados o mal dirigidos, dan respuesta a las solicitudes sencillas y envían los problemas que consideran críticos a la lista de correo de los desarrolladores[Mock00], [Mock02]. Cuando un problema de cualquier fuente ha sido solucionado, es buscado en los reportes de problemas para que sea incluido en el reporte de incidencias resueltas. El proyecto Apache contaba con una herramienta (BUGDB) para tal fin.

6 Amenazas a la Validez

La validez del SMS presentado en este trabajo se ve amenazada por incluir solamente artículos en inglés, pues todos los términos de búsqueda fueron definidos en este idioma. En este caso no existe riesgo de descarte de estudios primarios, al haber sido contrastados los resultados del proceso de selección con un experto en el área.

No podemos garantizar que todos los estudios primarios relevantes fueron seleccionados durante el proceso de búsqueda. Reducimos esta amenaza siguiendo las referencias en los estudios primarios. Dentro de la comunidad OSS hay una tendencia que aboga por licencias para los manuales y documentación que también sean "*open*", como por ejemplo *creative commons* y similares. Las fuentes consultadas no están soportadas mayoritariamente con este tipo de licencias, por lo que es probable que haya estudios fuera de dichas fuentes.

7 Conclusiones y Trabajos Futuros

Este trabajo de investigación reporta un estudio del proceso de mantenimiento seguido por la comunidad OSS, comparándolo con el prescripto por la ingeniería de software tradicional. Para conocer el proceso de mantenimiento OSS se ha realizado un SMS, cuyo objetivo era responder a la pregunta de investigación: ¿Qué actividades conforman los modelos de proceso OSS? Se encontraron un total de 22 estudios primarios, de los cuales el 73% contaba en sus procesos de desarrollo con actividades relacionadas con el mantenimiento. Estos estudios primarios sirven como punto de partida para realizar un análisis posterior de los procesos OSS y proponer un modelo del proceso seguido por esta comunidad.

La comunidad OSS no sigue los modelos y estándares prescriptivos de la ingeniería del software tradicional. El objetivo principal de esta comunidad es el soporte y mantenimiento de las funcionalidades existentes. No hay una distinción entre mantenimiento correctivo o evolutivo. El mantenimiento en la comunidad OSS puede ser definido como reinvención. Artefactos Web como listas de correos y e-mails son fundamentales en esta reinvención. Los proyectos OSS representan un paradigma alternativo al defendido durante mucho tiempo por la ingeniería del software tradicional.

Los proyectos OSS evolucionan a través de mejoras menores donde participan tanto usuarios como desarrolladores. En algunos proyectos OSS hay una prioridad esta-

blecida para la construcción de las nuevas funcionalidades, o para decidir cuales errores deben resolverse primero de acuerdo a una severidad. La corrección de errores es una tarea de introducción apropiada en un proyecto OSS, si se es una persona a quien le gustan los desafíos. La comunidad OSS también cuenta con actividades análogas a las prescriptas por el estándar IEEE 1074 [6], pero desarrolladas de manera diferente, como por ejemplo que los usuarios pueden contribuir con ideas o con código fuente para corregir los problemas reportados, o que cualquiera dentro de la comunidad pueda asignar prioridades a las mejoras del sistema software.

Dentro de nuestros trabajos futuros, está el realizar una investigación empírica, más de campo, en la que se estudien directamente los registros generados por la comunidad OSS, como listas de correos, wikis, etc. Además de estudiar los trabajos publicados en libresoft.es, http://2012.msrconf.org. Luego, definiremos un modelo de proceso de desarrollo OSS. Una vez definido este modelo, identificaremos un conjunto de técnicas de usabilidad apropiadas para su incorporación en las principales actividades de este modelo. Esta incorporación deberá tener en cuenta las características e idiosincrasias propias del desarrollo OSS, así como también la actividad del proceso donde será ubicada. Se pretende, por ejemplo, determinar cuál o cuáles técnicas de usabilidad pueden incorporarse en el grupo de actividades de mantenimiento.

Referencias

1. Acuña, S.T., Castro, J.W., Dieste, O., Juristo, N.: A Systematic Mapping Study on the Open Source Software Development Process. In: 16th International Conference on Evaluation and Assessment in Software Engineering (EASE'12), pp. 1-5 (2012)
2. Castro, J.W., Acuña, S.T.: Differences between Traditional and Open Source Development Activities. In: 13th International Conference on Product-Focused Software Development and Process Involvement (PROFES'12), Madrid, Spain, pp. 131-144 (2012)
3. Crowston, K., Wei, K., Howison, J., Wiggins, A.: Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. ACM Computing Surveys. 44 (2), Article 7 (2012)
4. Fuggetta, A.: Open Source Software: An Evaluation. Journal of System and Software. 66, 77-90 (2003)
5. Godfrey, M.V., Tu, Q.: Evolution in Open Source Software: A Case Study. In: International Conference Software Maintenance (ICSM'00), pp. 131-142. San José, CA (2000)
6. IEEE Std 1074:2006: IEEE Standard for Developing Software Life Cycle Processes. IEEE Computer Society (2006)
7. Kitchenham, B.A.: Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3. Technical Report S.o.C.S.a.M. Software Engineering Group, Keele University and Department of Computer Science University of Durham (2007)
8. Mockus, A., Fielding, R.T., Herbsleb, J.: A Case Study of Open Source Software Development: The Apache Server. In: 22st International Conference on Software Engineering (ICSE'00), pp. 263-272. Limerck, Ireland (2000)

9. Mockus, A., Fielding, R.T., Herbsleb, J.: Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology. 11(3), 309-346 (2002)
10. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic Mapping Studies in Software Engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08), pp. 71--80 (2008)
11. Potdar, V., Chang, E.: Open Source and Closed Source Software Development Methodologies. In: 26th International Conference on Software Engineering, pp. 105-109 (2004)
12. Scacchi, W.: Understanding the Requirements for Developing Open Source Software Systems. IEE Proceedings-Software. 149(1), 24-39 (2002)
13. Scacchi, W.: Free and Open Source Software Development Practices in the Computer Game Community. IEEE Software. 21(1), 59-67 (2004)
14. Scacchi, W.: Socio-Technical Interaction Networks in Free/Open Source Software Development Processes. In: Acuña, S.T., Juristo, N. (eds.) Software Process Modeling, pp. 1-27. Springer, New York (2005)
15. Scacchi, W., Jensen, C., Noll, J., Elliott, M.: Multi-Modal Modeling of Open Source Software Requirements Processes. In: First International Conference on Open Source Systems, pp. 1-8. Genova, Italy (2005)
16. Tian, Y.: Developing an Open Source Software Development Process Model Using Grounded Theory. Universidad of Nebraska – Lincoln, NB, USA, 143 pp. (2006)

Apéndice A: Estudios Primarios

Este apéndice contiene las referencias de los estudios primarios encontrados al realizar el SMS. A cada uno de los estudios primarios se le ha asignado un Nick para facilitar su referencia en el presente trabajo.

[Dinh05]: Dinh-Trong, T., Bieman, J.M.: The FreeBSD Project: A Replication Case Study of Open Source Development. IEEE Transactions on Software Engineering, 31, 481-494 (2005)

[Egan04]: Egan, S.: The Open Source Development Process. In Open Source Messaging Application Development: Building and Extending Gaim. Chapter 2, Apress, pp. 23-36 (2004)

[Erdo09]: Erdogmus, H.: A Process That Is Not. IEEE Software, 26(6), 4-7 (2009)

[Ezea08]: Ezeala, A., Kim, H., Moore, L.A.: Open Source Software Development: Expectations and Experience from a Small Development Project. In: 46th Annual Southeast Regional Conference on ACM-SE'08, pp. 243-246, Auburn, AL, USA (2008)

[Fitz06]: Fitzgerald, B.: The Transformation of Open Source Software. Forthcoming in MIS Quarterly, 30(3), 1-26. Including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics - 3840 LNCS (2006)

[Gurb06]: Gurbani, V.K., Garvert, A., Herbsleb, J.D.: A Case Study of a Corporate Open Source Development Model. In: 28th ACM International Conference on Software Engineering (ICSE'06), pp. 472-481 (2006)

[John01]: Johnson, K.: A Descriptive Process Model for Open-Source Software Development. Master. Thesis in Computer Science. Department of Computer Science. University of Calgary, 156 pp. (Online) http://sern.ucalgary.ca/students/theses/KimJohnson/thesis.htm (2001)

[Jorg01]: Jorgensen, N.: Putting It All in the Trunk: Incremental Software Development in the FreeBSD Open Source Project. Information Systems Journal, vol. 11(4), 321-336 (2001)

[Lell09]: Lelli, F., Jazayeri, M.: Community Support for Software Development in Small Groups: The Initial Steps. In: 2nd International Workshop on Social Software Engineering and Applications (SoSEA'09), pp. 15-22 (2009)

[Mart07]: Martin, K., Hoffman, B.: An Open Source Approach to Developing Software in a Small Organization. IEEE Software, 24, 46-53 (2007)

[Mock00]: Mockus, A., Fielding, R.T., Herbsleb, J.: A Case Study of Open Source Software Development: The Apache Server. In: 22st International Conference on Software Engineering (ICSE'00), pp. 263-272, Limerick, Ireland (2000)

[Mock02]: Mockus, A., Fielding, R.T., Herbsleb, J.: Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11(3), 309-346 (2002)

[Mong04]: Monga, M.: From Bazaar to Kibbutz: How Freedom Deals with Coherence in the Debian Project. In: 4th Workshop on Open Source Software Engineering-26th International Conference on Software Engineering (ICSE'04), pp. 71-75 (2004)

[Raym99]: Raymond, E.S.: The Cathedral and the Bazaar. In Cathedral and the Bazaar: Musing on Linux and Open Source by an Accidental Revolutionary, O'Really: Sebastopol, CA, pp. 19-64 (1999)

[Reis02]: Reis, C.R., Mattos Fortes, R.P.: An Overview of the Software Engineering Process and Tools in Mozilla Project. In: Workshop on OSS Development, Newcastle Upon Tyne, UK, pp. 162-182. (Online) http://opensource.mit.edu/papers/ reismozilla.pdf (2002)

[Scac04]: Scacchi, W.: Free and Open Source Development Practices in the Game Community. IEEE Software, 21(1), 59-66 (2004)

[Schw03]: Schweik, C.M., Semenov, A.: The Institutional Design of Open Source Programming: Implications for Addressing Complex Public Policy and Management Problems. Revista First Monday, 8(1), Chicago. (Online) http://firstmonday.org/issues/issue8_1/schweik/ index.html (2003)

[Seny04]: Senyard, A., Michlmayr, M.: How to Have a Successful Free Software Project. In: 11th Asia-Pacific Software Engineering Conference (APSEC'04). IEEE Computer Society, pp. 84-91, Bussan, Corea del Sur (2004)

[Simm03]: Simmons, G.L., Dillon, T.: Open Source Development and Agile Methods. In: 7th IASTED International Conference Software Engineering and Applications, pp. 523-527, Marina del Rey, CA, USA (2003)

[Vixi99]: Vixie, P.: Software Engineering. In Open Sources: Voices from the Open Source Revolution, Chapter 6, 1st Edition, de C. DiBona, S. Ockman, and M. Stone, (eds.) O'Reilly Press: Sebastopol, CA., pp. 91-100 (1999)

[Wynn04]: Wynn Jr., D.E.: Organizational Structure of Open Source Projects: A Life Cycle Approach. In: 7th Annual Conference of the Southern Association for Information Systems, pp. 285-290, Georgia (2004)

[Yama00]: Yamauchi, Y., Yokozawa, M., Shinohara, T., Ishida, T.: Collaboration with Lean Media: How Open-Source Software Succeeds. In: ACM Conference on Computer Supported Cooperative Work (CSCW'00), pp. 329-338 (2000)