Desarrollo Y Evaluación De Un Modelo De Procesos Para El Desarrollo De Software Libre Basado En El Modelo Bazar

Luis Galárraga, Alejandro Moreno, Verónica Macías
Departamento o Centro de la Universidad
Escuela Superior Politécnica del Litoral
Campus Gustavo Galindo Km. 30.5 Vía Perimetral, Guayaquil, Ecuador
lgalarra@fiec.espol.edu.ec, ghost482@yahoo.com, mmacias@fiec.espol.edu.ec

Resumen

El objetivo de este documento es proponer un modelo de procesos para desarrollar software libre basado en el estilo de desarrollo Bazar, el mismo que es ampliamente utilizado por proyectos FOSS (Free and Open Source Software) en la actualidad. Se han adoptado sus características más aceptadas y beneficiosas y se dejaron de lado aquellas prácticas que no son comunes y usualmente no ayudan al desarrollo en general. Adicionalmente se definieron una serie de métricas que permiten a los líderes de los proyectos, conocer el estado del proceso en cualquier instante de tiempo y tomar las decisiones pertinentes.

En este documento encontraremos también los pasos que tuvimos que seguir y los elementos necesarios para diseñar el modelo y para ponerlo en uso al desarrollar una aplicación para administración de seminarios en línea usando software libre. Al final del documento, se evalúa su validez a través de las métricas definidas durante su definición.

Palabras Claves: *software libre, modelo, desarrollo, open source, bazar*

Abstract

The purpose of this document is to propose a software development model for free software based on Bazaar development style which is widely used in FOSS (Free and Open Source Software) projects nowadays. We took Bazaar's most common and suitable practices and omitted those uncommon and not helpful. Additionally, we defined a set of metrics that allow project leaders to inquire about the state of the process in any moment in order to make the right decisions.

We will also found the followed steps to design the model and apply it through the development of an application for administration of online seminars or webinars. Finally, we assess the validity of the model using the metrics defined during its definition.

Keywords: *free software, model, development, open source, bazaar*

1. Introducción

Las tecnologías FOSS (Free Open Source Software) brindan a los países una gran oportunidad para reducir la brecha digital. Algunos países, entre estos Brasil [1], Francia [2], España [3] y Alemania [4] están aprovechando las ventajas del potencial del FOSS a través de leyes que regulan el uso de este tipo de tecnologías en las instituciones públicas y en la educación. A causa del creciente interés, la investigación de diversos aspectos (técnicos, sociales y económicos) del FOSS se ha intensificado en los últimos años.

A pesar de los beneficios que brinda el uso del FOSS, existen ciertos inconvenientes para su adopción. Uno de estos inconvenientes, es que actualmente no existe un proceso de desarrollo formalmente definido que se pueda seguir cuando se desea emprender un proyecto de FOSS; y debido a la falta de este modelo estandarizado, el desarrollo de software usando características del estilo Bazar no es visto como una alternativa seria o viable. Por otra parte, las compañías a menudo apuntan a definir su propio modelo de procesos más o menos basado en los modelos tradicionales estudiados por la Ingeniería de Software. Dichos modelos caen dentro de lo que Eric. Raymond [5] denominó modelos estilo Catedral. En general, estos modelos de desarrollo permiten a los líderes de proyectos estimar tiempos y costos; pero no son aplicables al software libre.

Por esa razón, el presente artículo muestra los resultados obtenidos durante la definición y evaluación de un modelo de proceso de desarrollo estilo Bazar.

2. Acciones realizadas para diseñar y evaluar el modelo definido.

2.1. Diseño del modelo de proceso de desarrollo

Para diseñar el modelo de desarrollo se tomaron las prácticas más relevantes del modelo Bazar. Es vital resaltar que una de las principales y más cuestionadas características de este modelo, es la ausencia de un proceso paralelo de control de calidad [6], el cual sí fue incluido en el modelo de procesos definido. El documento de definición del modelo puede ser consultado en [7].

La definición del proceso de control de calidad, implicó la selección de un conjunto de métricas del proceso y del producto para evaluar la salud del proyecto; siendo la salud del proyecto una medida de la probabilidad de éxito del software construido. La palabra éxito es subjetiva, sin embargo, en software libre un proyecto es exitoso si posee una comunidad activa de miembros que sigue aportando al proyecto, produciendo como consecuencia una mejora continua del mismo a través del tiempo. La tabla 1 expone las métricas que fueron escogidas para evaluar esta instancia del modelo con sus respectivas unidades.

Tabla 1. Métricas escogidas en la aplicación del modelo

Métrica	Unidades
Momentum o frecuencia de liberaciones	Liberaciones por mes
Número de correos enviados a la lista por unidad de tiempo	Total de correos en el período de análisis
	Promedio de correos por día
	Total de mensajes de foro en el período
	Promedio de mensajes de foro por día
Frecuencia de conflictos en el servidor de control de versiones	Conflictos por mes
Tasa de *commits* o aportaciones erróneas por unidad de tiempo.	Tasa de commits erróneos por mes
Tiempo de vida de un *bug*	Promedio del tiempo de vida en días laborables de los bugs resueltos en el período
	Promedio del tiempo de vida en número de liberaciones de los bugs resueltos en el período
Retraso en la entrega de liberaciones	Días laborables entre la fecha programada y la fecha real de la liberación
Número de bugs reportados por *release*	Número de bugs reportados y aceptados entre dos liberaciones

Es necesario acotar que el modelo en muchos casos no propone las unidades a usar en el control del proceso, pues éstas dependen exclusivamente del proyecto de software.

Las métricas expuestas corresponden a métricas del proceso y del producto; y todas las métricas que incluían unidades con períodos de tiempo en días, emplearon tan solo días laborables, por lo que se omitieron fines de semana y feriados.

2.2. Evaluación del modelo

Una vez definido el modelo y sus métricas, el siguiente paso fue evaluarlo; y para ello se decidió desarrollar una aplicación FOSS para manejo de seminarios en línea, siguiendo el modelo diseñado. Esta aplicación, bautizada como openASEL [8], se basó en la plataforma AccessGrid; esto con el fin de seguir la práctica recomendada para proyectos FOSS

de iniciar un desarrollo basado en aplicaciones ya existentes.

Paralelamente a la selección de la aplicación a desarrollar, se probó y evaluó un grupo de herramientas de soporte a fin de facilitar la recolección de las métricas para el proyecto propuesto. Cabe recalcar que el modelo no recomienda en ningún momento el uso de una herramienta en particular, por que la viabilidad de las mismas depende exclusivamente del proyecto de software donde vayan a ser utilizadas.

Como paso siguiente se escogió los desarrolladores para el proyecto. Aquí debemos acotar que inicialmente se tenía pensado trabajar con desarrolladores de entidades ubicadas en diferentes locaciones geográficas, para simular de manera adecuada un ambiente de desarrollo FOSS. Sin embargo, debido a la falta de voluntarios de las entidades que inicialmente manifestaron su interés y a las restricciones de tiempo impuestas para el proyecto, el grupo de desarrollo estuvo conformado por 2 tesistas y 4 integrantes de la comunidad de software libre KOKOA, todos estudiantes de la ESPOL. A estos desarrolladores se los capacitó para trabajar con el modelo y con las herramientas de soporte seleccionadas.

Luego se dio inicio al desarrollo del proyecto openASEL, y a medida que este avanzaba se realizaban las mediciones respectivas. Los datos fueron recolectados en períodos de tiempo entre 5 y 15 días donde se analizaron las métricas que tenían validez durante ese período.

3. Análisis de resultados

Los indicadores más precisos de la actividad de los miembros de un proyecto son los canales de comunicación del mismo, para nuestra instancia, la lista de correos y el foro. La figura 1 muestra la actividad de la lista de correos en el período de desarrollo del proyecto.

La lista de correos fue empleada como el principal medio de comunicación del proyecto. Un vistazo a la figura, nos permite notar una tendencia a la baja en su actividad durante el proceso, con un repunte al final del período de análisis. Ese comportamiento es predecible, pues al inicio del proceso se necesitó una intensiva comunicación para familiarizarse con el proceso de desarrollo. Una vez terminada esta etapa, cada grupo de trabajo se enfocó en sus actividades con lo cual la actividad de envío de correos decreció, para luego incrementarse al final del período, cuando se estableció que se debía realizar una liberación parcial del software. Este último hecho revela una característica de esta métrica: un descenso en su valor no necesariamente implica un decrecimiento real de la actividad del proyecto ni en su salud. Para emitir juicios sobre la real actividad, es imprescindible tener información sobre el contexto en el que fueron tomados los datos. Para proyectos de mayor trayectoria con múltiples liberaciones es de esperarse que esta métrica presente repuntes en las fechas cercanas a eventos destacados como la publicación de una nueva versión del software.

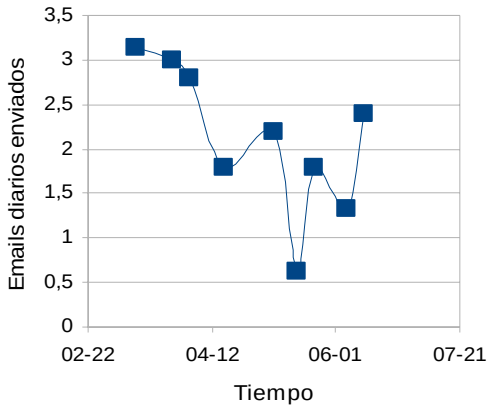

Figura 1. Gráfico de la actividad en la lista de correos.

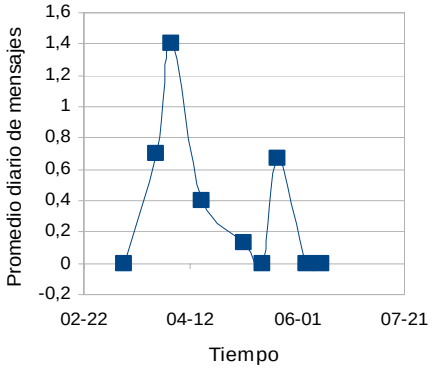

Figura 2. Gráfico de la actividad en el foro de discusión

A diferencia de la lista de correos, la actividad en el foro de discusión fue una métrica poco contundente para nuestra instancia del modelo. La figura 2, muestra actividad en las primeras fases del desarrollo con un descenso sostenido y valores nulos en ciertos períodos. El foro de discusión fue utilizado como medio para discutir temas técnicos muy puntuales que tuvieran acceso público al considerarlos útiles para los futuros usuarios de la aplicación. En proyectos FOSS con grandes

comunidades de usuarios, la actividad en los foros es intensa, pues son los miembros ajenos al desarrollo quienes solicitan ayuda a través de este tipo de canales.

Además de analizar la actividad global del proyecto, hemos considerado necesario desglosarla por usuario. La razón es destacar una característica que es muy común en los procesos estilo bazar no siendo éste la excepción a la regla. La mayor actividad en los canales de comunicación la concentran unos pocos desarrolladores, en la mayoría de los proyectos, sus líderes. La figura 3 muestra la distribución de la actividad total en la lista de correos por usuario y en repositorio al cabo de la última revisión al momento de escribir este reporte.

Como se ha observado en los resultados mostrados hasta aquí, unos pocos usuarios concentran la mayor actividad en los canales de comunicación y en el repositorio; sin embargo esto no refleja una mala salud en el proyecto, ya que esta situación es muy común en los proyectos FOSS [7]. Un problema que se identificó en el desarrollo fue que los desarrolladores que no eran líderes del proyecto no acostumbraban a indagar suficiente sobre las tareas que tenían que hacer y eran los líderes quienes debían averiguar si todo estaba claro (lo cual en ocasiones no era cierto). Al inicio también afectó el hecho de que el ambiente fuera simulado y los desarrolladores pudieran tener contacto de forma relativamente fácil, evitando el uso de los canales de comunicación cuando era realmente más rápido hacerlo por este medio. Esto fue corregido con el paso del tiempo gracias a los reportes de métricas realizados periódicamente.

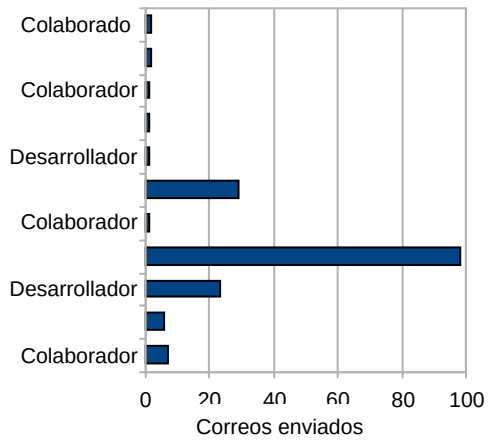

Figura 3. Gráfico de la actividad en la lista de correos por usuario

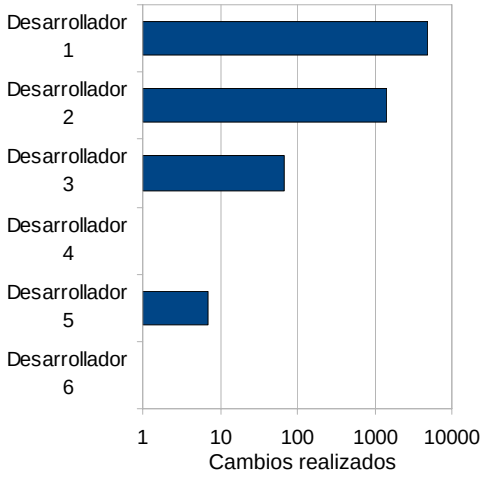

Figura 4. Gráfico de la actividad en el repositorio por usuario

En la figura 5 se muestra la tasa de aportaciones erróneas en el tiempo. Como se puede observar esta tasa no tuvo un mayor crecimiento; pero esto no significa que todo el código estaba perfecto, ya que los resultados dependen mucho de un factor clave que no ha sido tomado en cuenta hasta ahora: el tamaño de la comunidad de desarrolladores. Nuestra comunidad simulada contaba con 6 desarrolladores, con lo cual la probabilidad de generar conflictos por trabajar en una misma sección del proyecto es muy baja. Por otra parte, todos los desarrolladores carecían de experiencia en desarrollo FOSS, factor que si bien trató de ser contrarrestado con las jornadas de capacitación, no evitó que cometieran un par de errores al momento de interactuar con el repositorio y las herramientas de gestión.

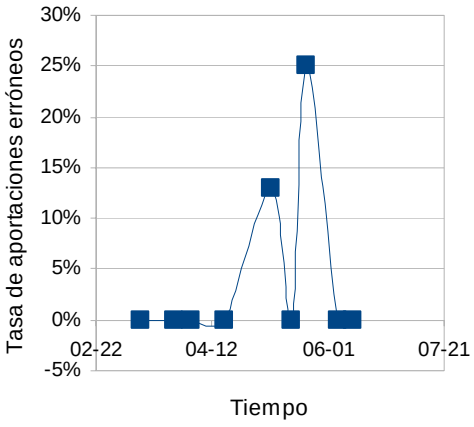

Figura 5. Gráfico de las aportaciones en el repositorio por desarrollador

En relación al resto de métricas, la presencia de datos obtenibles fue escasa por factores imputables a la instancia. Con un período de desarrollo de 4 meses basado en una herramienta que nadie conocía, era predecible. El desarrollo no pudo arrojar ni una versión estable del software, solo un par de versiones pre-alpha de acuerdo a la decisión de la comunidad. Esto se debió a un sinnúmero de factores externos dentro del cual excluimos apatía de parte de los desarrolladores. El principal fue la complejidad de la herramienta base, en este caso, la plataforma AccessGrid. Aquí vale hacer una aclaración importante, pues recibimos mucha ayuda de parte de algunos miembros de esta comunidad, al igual que el ofrecimiento de una base de código que ellos habían implementado y que permitiría a nuestro proyecto interactuar con facilidad con AccessGrid. Implementar dicho módulo habría significado algunas semanas adicionales de investigación, diseño e implementación que nuestras restricciones de tiempo nos impedían asumir. Lamentablemente, la ayuda nunca llegó con lo cual fue imposible generar una versión completamente estable y funcional de la aplicación.

4. Conclusiones y mejoras a futuro

El modelo expone una serie de prácticas recomendadas para garantizar pleno conocimiento del estado del proyecto en cualquier instante de tiempo; sin embargo no ofrece ninguna recomendación de cómo solucionar un caso de apatía por parte de los miembros y colaboradores del proyecto, pues dicha parte es muy subjetiva y es una debilidad inherente al desarrollo estilo Bazar, donde quienes colaboran, en muchos casos, lo hacen de forma voluntaria. La solución a este tipo de inconvenientes es la elaboración de planes de riesgo que permitan definir un plan B ante cualquier inconveniente. Esta es una tarea que no se encuentra para nada reñida con la esencia del desarrollo Bazar por lo que debe ser considerada por quienes apliquen este modelo.

No todas las métricas definidas en el modelo sirven para todas las instancias del mismo. Comunidades con pocos desarrolladores pueden no requerir de métricas como la tasa de conflictos, así como comunidades muy jóvenes pueden no necesitar de las métricas relacionadas a la actividad en la liberación de versiones del software. Para estos casos, puede ser útil recurrir a otras métricas que si bien no han sido definidas explícitamente en el modelo, pueden resultar muy útiles y son derivables de las ya definidas.

Para finalizar, se puede afirmar que el modelo propuesto es válido en gran parte, mas no en su totalidad. Prueba de esto es el hecho que no logramos conseguir nuestra meta inicial, liberar un prototipo estable. Aunque se ha mencionado anteriormente que esto se debió en gran parte a motivos externos al modelo o a problemas inherentes al desarrollo voluntario de comunidades de código abierto, no es razón válida para decir que el modelo no tuvo su aportación en el retraso. Esto significa que aunque el modelo es un gran paso hacia la estandarización del proceso de desarrollo de software libre, se debería probar de forma más extensiva para mejorar sus defectos y añadir las mejoras posibles.

5. Referencias

[1] Brasil y la ofensiva Linux. (2005, Junio 3). Recuperado Junio 18, 2008 de http://news.bbc.co.uk/hi/spanish/business/newsid_4606000/4606701.stm

[2] French National Assembly switches to Linux. (2006, Noviembre 26). Recuperado Junio 18, 2008, de http://www.pcadvisor.co.uk/news/index.cfm?newsid=7687

[3] España, una potencia del software libre. (2004, Abril 16). Recuperado Junio 18, 2008, de http://www.laflecha.net/canales/softlibre/20040404161/

[4] Alemania es el país con mayor uso de Software Libre. (2007, Junio 25). Recuperado Junio 18, 2008, de http://www.mastermagazine.info/articulo/11901.php

[5]Raymond, Eric S. (2001) The Cathedral & the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly

[6] Fogel, K. (2005). Producing Open Source Software. How to Run a Successful Free Software Project. O'Reilly

[7] Galárraga L., Moreno A. MOCCA-Modelo Controlado para Código Abierto. Recuperado Noviembre 24, 2008 de https://proyectossw.espol.edu.ec/frs/download.php/1/MOCCA.pdf

[8]openASEL-Administración de Seminarios en Línea. http://proyectossw.espol.edu.ec/websites/openasel