Un enfoque práctico para la elección y adecuación de Software Open Source de Aplicación

Sandra Casas y Eugenia Márquez

Plan de Acción de Sistemas (PAS) - Universidad Nacional de la Patagonia Austral
Lisandro de la Torre 860. CP 9400. Río Gallegos. Santa Cruz. Argentina
Tel/Fax: +54-2966-442370.
E-mail:coihue@unpa.edu.ar

Resumen: Este trabajo presenta un enfoque práctico para la elección y adecuación de software Open Source de aplicación a contextos específicos. La propuesta esta inspirada en metodologías ágiles de desarrollo de software y toma de éstas un conjunto de características que lo hacen práctico, fácil y a la vez desafiante. Este método ha sido aplicado a un proyecto real, cuyo software esta finalizado y en etapa de operación.

1. Introducción

El movimiento Open Source (OSS) y el Software Libre aportan principalmente soluciones para el desarrollo y soporte de sistemas software (sistemas operativos, servidores, lenguajes y entornos de programación, gestores de bases de datos, etc.). Por ejemplo Linux[1], MySQL[2], PHP[3], Java[4], Apache Web Services[5], Herramientas Eclipse[6], etc., son herramientas OSS o libres de estas características muy utilizadas actualmente. También pueden obtenerse herramientas de apoyo a los distintos procesos de ingeniería de software[1]. Este tipo de software favorece la viabilidad técnica y económica del desarrollo de proyectos de software. La calidad y garantía de estos productos de software esta al mismo nivel que los comerciales y en algunos casos hasta se considera superior. Ciertas características del OSS, como la gratuidad y libre licenciamiento, rápida y fácil accesibilidad, aceptable calidad y buen rendimiento, además de la amplia difusión e información, hacen que estas iniciativas crezcan y se afiancen en forma contundente. Sin dudas el OSS se ha convertido en una de las mejores alternativas desarrollar software de aplicación de menor costo.

Una mirada diferente debe hacerse sobre el OSS de aplicación. Un software de aplicación resuelve problemas reales esencialmente referentes a sistemas de información, que involucran procesos de gestión de datos tales como la liquidación de los sueldos del personal de una empresa, la gestión de cuentas de un banco, la gestión de stock de un depósito, las reservas y ventas de pasajes de un transporte de pasajeros, etc. Este tipo de aplicaciones se componen de un conjunto de módulos funcionales, que actualizan una base de datos relacional en forma constante y resultan ser operaciones vitales los reportes, las validaciones de los datos de entrada, el cálculo automático de operaciones matemáticas, financieras y/o estadísticas sobre grandes volúmenes de datos, etc. A este tipo de problemas también han llegado las soluciones OSS, y es así que se encuentran disponibles en la red distintos productos para el mismo tipo de sistema. Sin embargo, ocurre que cada organización tiene sus propias reglas de negocios y particularidades, lo cual hace improbable que un OSS de aplicación cubra en forma total y exacta los requisitos específicos de dicha organización. Al tratarse de OSS se dispone del código fuente, por lo tanto puede adecuarse, personalizarse, mejorarse para cumplir con la totalidad de los requisitos. En este escenario, el primer problema que se debe resolver cuando existe más de una alternativa OSS, es la elección de uno de ellos. Cómo se toma esta decisión, cuando el factor costo del producto ya no es un determinante. El otro punto interesante para analizar se (luego que se ha seleccionado un determinado OSS de

[1] Open Source Software Engineering Tools: www.tigris.org

aplicación, cuya funcionalidad se ajusta a lo requerido en parte), refiere a los métodos y/o técnicas adecuados a emplear para adaptar, modificar o agregar la funcionalidad que resta. En este último sentido no existen formalmente métodos y técnicas para OSS de aplicación.

Un problema real en la adopción de un OSS de aplicación y su adecuación, nos ha permitido ensayar un enfoque práctico que condujo a resultados positivos. El método propuesto cumple con los dos objetivos enunciados: la elección del OSS de aplicación y su adecuación.

El enfoque propuesto esta fuertemente influenciado por las metodologías ágiles [7][8][9][10] y se cumplen con varios principios del manifiesto ágil [11]. Se ha tratado de mantener ciertas características de estas metodologías ya que resultan adecuadas a esta problemática como ser: equipos de trabajo pequeños, planificación consensuada, fuerte orientación al código y a la prueba, etc.

En el presente trabajo explicamos el enfoque utilizado en esta problemática. En la Sección 2 describimos como se realiza la Evaluación y Elección del OSS de aplicación utilizando este enfoque; en la Sección 3 se explica el Proceso de Desarrollo, en la Sección 4 se citan una serie de recomendaciones y prácticas; en la Sección 5 se describe brevemente la experiencia realizada y en la Sección 6 se presentan las conclusiones.

2. Evaluación y Elección de OSS de aplicación.

El punto de partida implica que existe más de una alternativa OSS para un determinado problema. Este paso se excluye en aquellos casos que se n el caso q Considerando que éstos productos cumplen con ciertos requisitos, pero no todos, es necesario evaluar cual de éstos es el más conveniente. Se buscará aquel software que se aproxime más a la realidad, ya que demandará menos esfuerzo y tiempo de adecuación.

Se comienza elaborando una lista de los requisitos que el futuro software debe cumplir. Esta lista debe ser lo más completa posible. Cada ítem debe expresarse en forma precisa y concisa. En primer lugar irán los requisitos funcionales, pero deben incluirse también requisitos de capacidad, interfase, operación, seguridad y transportabilidad al menos.

El segundo paso consiste en asignar prioridades a los ítems de la lista. Cada ítem tiene una prioridad alta (A), media (M) o baja (B). Los criterios para la asignación de prioridades pueden diferir, pero lo mas acertado será dar mayor prioridad a aquellos items que para el cliente son más importantes.

El último paso consiste en confeccionar una tabla en la cual se compare el nivel de satisfacción de los requisitos por los distintos OSS de aplicación que se están analizando. Se utilizan tres niveles de cumplimiento: cumple totalmente (“C”), cumple parcialmente (“P”) y no cumple (“N”).
El resultado es como se indica en la Tabla 1.

Tabla 1: Comparación de OSS de aplicación

Prioridad	Requisitos	OSS_1	OSS_2	OSS_3
A	R_1	C	P	P
M	R_2	N	P	C
A	R_3	P	C	C
B	R_4	N	P	C
B	R_5	C	N	P
M	R_6	N	P	N

Utilidad y Objetivos de la Tabla
El primer objetivo de la tabla es servir a la selección del software que se va a adoptar. En esta decisión participa el cliente. La decisión podría estar basada en elegir aquel que cumpla la mayor cantidad de ítems de prioridad A en forma total o parcial, por ejemplo. El segundo objetivo de la tabla es servir como guía del proceso de desarrollo, por eso su completitud y exactitud son de vital importancia.
La tabla debe construirse aun si existe solo un OSS para evaluar, en este caso solo servirá para el segundo objetivo. Al finalizar esta actividad

se tiene un conocimiento bastante aproximado del trabajo a realizar.

3. Proceso de Desarrollo

El proceso de desarrollo tiene por objeto que el OSS de aplicación elegido cumpla con todos los ítems de la tabla planteados. Por eso, la tabla se convierte en la guía de un proceso de desarrollo iterativo. El objetivo de cada iteración es obtener una versión del producto, que sea instalable y puesta a la operación del usuario. Esta es una versión incompleta, pero que tiene sentido para el usuario. Luego de cada iteración la versión instalable se va completando, hasta concluir con la versión final.

3.1 Planificación Global Inicial

Antes de comenzar el proceso iterativo, se realiza una planificación global que estima el tiempo que llevará el proyecto. De acuerdo a la cantidad y complejidad de los requisitos y el equipo de trabajo se calcula una cantidad de iteraciones mínimas y máximas. Es decir, se hace una planificación optimista y una pesimista. La diferencia entre ambas no puede superar las 4 cuatro iteraciones. Entre estos márgenes deberá suponerse la duración del proyecto. Las capacidades, experiencias del equipo de trabajo son fundamentales en este sentido.

3.2 Iteraciones

Una iteración comienza con la elección de uno o más ítems de la lista de requisitos de prioridad ALTA con nivel de cumplimiento P o N. Los ítems escogidos deben dirigirse al mismo objetivo funcional, en función de dar más valor y sentido a la futura versión.

Una iteración lleva de 7 a 12 días, dependiendo de los ítems escogidos y el equipo de desarrollo. Las primeras iteraciones serán las más largas y/o resolverán menos ítems de la lista. Progresivamente los desarrolladores conocerán las características internas del OSS de aplicación (diseño arquitectónico, código fuente, estructuras de datos) y la producción crecerá luego de pasar las primeras iteraciones. Al finalizar una iteración, no siempre se tendrá una nueva versión. También puede trabajarse sobre la mejora y optimización de una versión anterior.

En cada iteración se realizan cuatro actividades principales: planificación, desarrollo, pruebas e instalación, como se esquematiza en el Figura 1.

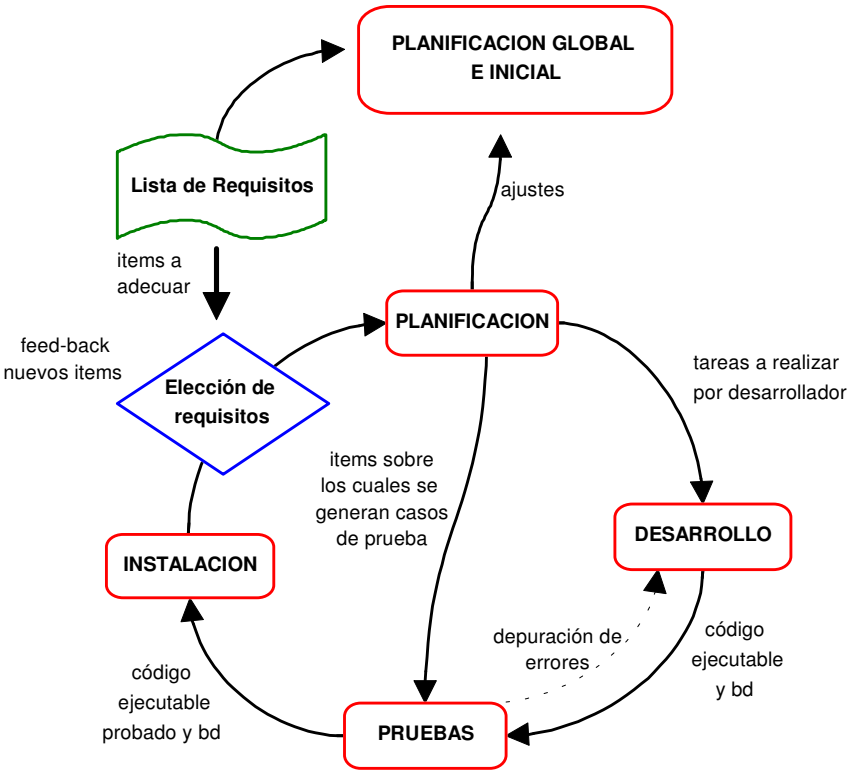

Figura 1: Proceso de desarrollo – Actividades de una iteración.

Planificación de Iteración
Los ítem de la lista de requisitos elegidos para la iteración que se inicia, son analizados por el equipo de trabajo, en cuanto a la complejidad y tareas que demandan su adecuación. Por ejemplo deben especificarse los datos que faltan en las interfases, operaciones de validación nuevas o innecesarias o, nuevos informes, etc. Cada desarrollador toma notas de la tarea que debe realizar y en forma consensuada el equipo estima el tiempo que requiere la iteración. Si la estimación supera los 12 días, significa que se han tomado demasiados ítems de requisitos, en este caso deben quitarse ítems.

Desarrollo
El desarrollo consiste esencialmente en la modificación de código fuente, interfaces y estructura de la base de datos. Los programadores analizan e identifican los componentes que deben ser modificados y en consecuencia agregan, modifican o eliminan código. Realizan pruebas individuales que les permitan asegurar la correctitud de los cambios que han introducido.

Pruebas
En forma simultánea al desarrollo, se generan los casos de prueba. Las pruebas que se ejercitan son de tipo funcionales. Están dirigidas a probar el correcto funcionamiento de las interfaces, actualización de la base de datos, verificación de datos de entrada, cálculo de resultados, generación de informes, etc. Aquí las pruebas pueden dividirse en (a) pruebas de los ítems resueltos en la iteración, y (b) pruebas para garantizar que el resto de la aplicación continua funcionando correctamente.
Preferentemente es actividades del responsable del proyecto, cuya experiencia y conocimientos indica que propondrá buenos casos de prueba.
Al finalizar el desarrollo en primer lugar se ejercitan las pruebas. Cuando todas las pruebas son superadas se continúa con la instalación.

Instalación
En esta fase se realizan actividades que permitan al cliente operar la versión del sistema producto de la iteración. Se instala el sistema o nuevos componentes. Se instruye al cliente en su uso, dado que se esta instalando una función o subfunción no debería llevar demasiado tiempo. Por último se realizan las migraciones o importaciones de datos necesarias.

4. Prácticas y Recomendaciones

Codificación
La codificación es la tarea principal de desarrollo. El propósito es que el código modificado se funda en el código original, los lineamientos que se recomiendan seguir son: (a) adoptar el estilo de codificación original, (b) modificar, agregar y remover solo el código que sea necesario, (c) en una misma iteración dos desarrolladores no trabajan sobre el mismo código fuente, formularios o tablas. Aplicar buenas practicas de programación [12][13][14].

Documentación
No esta entre los objetivos del proceso de desarrollo reconstruir la documentación faltante. Solo se reconstruye aquella documentación que se considera necesaria. Por ejemplo un diagrama entidad relación de la base de datos resulta muy útil, al igual que un diagrama de clases que indique la relaciones entre las mismas y su funcionalidad básica. La documentación más importante es el código fuente [6], por eso su construcción debe ser bien realizada. Otro aspecto que debe ser bien gestionado es el control de versiones. La catalogación de todas las versiones que se van produciendo a lo largo del proyecto debe indicar al menos, para cada una de las mismas, que ha cambiado respecto de la anterior (código y base de datos).

Equipo de trabajo
El equipo de trabajo esta conformado por un grupo de 3 a 5 personas, todas con un perfil preponderante de programación. El integrante de mayor experiencia es el responsable del proyecto. Su principal función es la de mediar y consensuar el trabajo. Aquellos factores que afectan a todo el equipo, como la asignación de actividades y estimaciones es preferente que se realice de forma concensuada, al igual que la estimación del tiempo de cada iteración.

Validación y Verificación de requisitos
La pronta instalación de cada versión, tiene como ventaja que el cliente rápidamente entra en contacto con una parte del software. Esta interacción temprana seguramente producirá un feed-back, deseable y positivo, aun cuando

implique modificar el mismo código nuevamente Luego de la instalación de las primeras versiones, surgirá información de posibles errores u omisiones en los requisitos que se transformarán en nuevos items para la lista y serán abordados en la próxima iteración. La idea es modificar el código tan pronto como se produjo, y no dejar pasar mucho tiempo.

5. Experiencia Realizada

El método descrito se aplicó en un proyecto que tenia por objetivo reemplazar 3 software de aplicación que se utilizaban para la gestión bibliotecaria de una institución educativa de nivel superior. El proyecto surgió ante la necesidad de unificar la base de datos, procedimientos y ofrecer servicios en web, además de agregar mayor control y seguridad en las operaciones. Las aplicaciones OSS que se analizaron fueron Emilia [15], MyPHPLibrary [16] y OpenBiblio [17]. Resultando este último el escogido. Las principales funcionales del sistema bibliotecario estaban soportas en OpenBiblio, pero fue necesario ajustar detalles en casi todas ellas. Algunos de estos ajustes fueron extremadamente simples, por ejemplo hacer la traslación al español de todos los formularios y mensajes. Otros ajustes requirieron mayor esfuerzo, por ejemplo el añadir el registro de actividades de las operaciones de mayor relevancia. El equipo estuvo conformado por 3 integrantes. La primera versión del software se instaló el 28/09/06 y desde entonces semanalmente se actualizó la misma. La tarea que llevo mas tiempo y esfuerzo, fue la migración de la bases de datos del catalogo del material bibliográfico. En este caso, la diferencia de formatos y la falta total de normalización en los datos originales requirieron arduos procesos para convertirlo en una base de datos relacional. El sistema se encuentra actualmente en producción.

6. Conclusiones

En este trabajo se ha presentado brevemente un enfoque práctico para seleccionar y adaptar OSS de aplicación. Este enfoque ofrece las siguientes ventajas: es un método de trabajo ordenado, facilita la vuelta atrás, es apropiado para obtener soluciones rápidas y de bajo costo, es fácil de aprender y aplicar, en todo momento se sabe que esta terminado y que resta por hacer. Sus principales características: Iteraciones cortas, equipos de trabajo pequeño, Planificación consensuada, fuerte orientación a la implementación y prueba, planificación orientada por prioridades.

El trabajo futuro se refiere fundamentalmente a dar a conocer este incipiente método para que sea aplicado a otros proyectos de modificación de OSS de aplicación. La puesta en práctica permitirá encontrar errores, defectos y omisiones que progresivamente ayudarán y contribuirán a completar, mejorar y fortalecer el método original.

El presente trabajo fue parcialmente financiado por la Universidad Nacional de la Patagonia Austral, Santa Cruz, Argentina.

Referencias

[1] Home Page de Linux: http://www.linux.org/
[2] Home Page de http://www.mysql.org/
[3] Home Page de http://www.php.net/
[4] Home Page de http://java.sun.com/
[5] Home Page de http://www.apache.org/
[6] Home Page de http://www.eclipse.org/
[7] Beck K., "Extreme Programming Explained: Enhance Change", Addison-Wesley, Reading MA, 2.000.
[8] Cockburn A., Highsmith J.: Desarrollo de Software Agil Series, Addisson Wesley Professional.
[9] Cockburn A. "Crystal Clear: A Human-Powered Methodology for Small Teams". Addison-Wesley. 2004.ISBN-10: 0201699478
[10] Home page de Scrum: http://www.controlchaos.com/
[11] Manifiesto para el desarrollo de software Agil http: //www.agilemanifesto.org
[12] Hunts A., Yhomas D., "The Pragmatic Programmer: From Journeyman to Master"
[13] Fowler M., "Refactoring: improving the design of existing code. Addison-Wesley, 1999, ISBN 0-201-48567-2.
[14] McConnell S. "Code Complete" 2 Edition, Microsoft Press 2004, ISBN 10-0735619670.
[15] Home page de Emilda http://www.emilda.org/
[16] Home page de phpmylibrary http://sourceforge.net/projects/phpmylibrary
[17] Home page de Openbiblio: http://obiblio.sourceforge.net/