Modelo para el Aseguramiento de Calidad en el Desarrollo de Software Libre

Cenditel, Mayo 2011

Licencia de Uso

Índice de Contenido

Modelo para el Aseguramiento de Calidad en el Desarrollo de Software Libre

El desarrollo de aplicaciones bajo la filosofía del software libre ha tomado una importancia significativa en los últimos años, dada la calidad de las aplicaciones que han sido desarrolladas bajo esta filosofía. Cabe destacar que en el ámbito de software libre, al igual que en el ámbito del desarrollo propietario, la calidad esta determinada por la mejora continua del software, a fin de cumplir con los requerimientos establecidos por los usuarios.

La calidad de las aplicaciones de software depende fundamentalmente del proceso de desarrollo que se siga en la construcción de las mismas. Por tanto, las metodologías y métodos de desarrollo constituyen elementos determinantes para lograr el cumplimiento de los requerimientos de calidad de las aplicaciones de software.

A pesar de la calidad del software libre existen fuertes críticas respecto a los procesos de desarrollo que se siguen en la construcción de este tipo de aplicaciones. Estas críticas están dirigidas fundamentalmente a que en la mayoría de los proyectos de software libre, con excepciones de algunos proyectos como *Debian*, *Ubuntu*, entre otros, no se siguen metodologías o métodos formales de desarrollo que faciliten y promuevan prácticas de documentación del software. Cabe destacar que las prácticas de documentación representan un factor determinante para los procesos de apropiación del software, por tanto, para el cumplimiento de dos de las libertades del software libre, a saber, *estudio* y *mejora* del software. La documentación del software no solo facilita el uso del mismo, sino que constituye la base para realizar cualquier estudio de éste con fines de entender su funcionamiento o mejorarlo.

Es de destacar que a pesar de las críticas mencionadas, por lo general, las aplicaciones desarrolladas en software libre han logrado cubrir con mayor satisfacción los requerimientos de calidad de los usuarios que las aplicaciones desarrolladas en software propietario. Esta particularidad del software libre, en cuanto a la calidad de sus aplicaciones, ha llevado a muchas organizaciones al estudio del proceso de desarrollo en el ámbito del software libre, ello con la finalidad de determinar las prácticas características de este tipo de desarrollo, pues la calidad del software parece deberse a dichas prácticas.

La Fundación Cenditel considerando lo planteado en los párrafos anteriores y, en base a sus necesidades como centro de desarrollo de tecnologías libres, entre ellas desarrollo de software, se plantea la elaboración de un modelo para el aseguramiento de calidad en el desarrollo de software libre, en el cual se busca plantear un modelo estándar que facilite la mejora continua del proceso de desarrollo de software, con lo cual no sólo se podrá mejorar la calidad de las aplicaciones que se desarrollen sino que se contribuirá con los procesos de apropiación de estas aplicaciones.

Para que el modelo pueda contribuir a la mejora continua de los procesos de desarrollo de software libre, así como a la mejora de las aplicaciones desarrolladas y a la facilitación de los procesos de apropiación de estas aplicaciones, se propone elaborar un modelo híbrido en el cual se combinen prácticas estándar de la ingeniería de software con prácticas características del software libre.

A continuación se presenta un análisis de las prácticas características del desarrollo de software libre y de la ingeniería del software, a fin de determinar las prácticas en base a las cuales se fundamentará el modelo propuesto.

1. Prácticas Características del Desarrollo de Software Libre

El desarrollo de software libre se caracteriza fundamentalmente por la adopción de prácticas que facilitan el desarrollo colaborativo de aplicaciones de software entre desarrolladores que, por lo general, no se ubican en un mismo lugar.

La calidad en las aplicaciones de software libre depende básicamente de ciertas características presentes en sus prácticas de desarrollo, las cuales no se encuentran, por lo general, en las prácticas de desarrollo de software privativo. A continuación, mencionaremos algunas características de las más importantes.

- Publicación del código fuente.

  La publicación del código es una de las principales ventajas y prácticas características del software libre, pues no solo permite que otras personas puedan utilizar el software, sino que facilita y promueve el reporte y corrección de errores en el software por parte de personas internas o externas a la comunidad de desarrollo. Es importante destacar que en los desarrollos de software libre se trata, en la medida de lo posible, de liberar frecuentemente prototipos o versiones del código, lo cual facilita que muchas personas (entre ellas usuarios y otros desarrolladores) puedan participar en las pruebas y mejoras de éste. Esta participación facilita el reporte de mayor cantidad de errores que los que podrían encontrar los desarrolladores del software, además de facilitar las propuestas de mejoras en términos de modificaciones o desarrollo de nuevas funcionalidades.

- Comunidad de desarrollo.

  La conformación de una comunidad en torno al desarrollo del software permite contar con una variedad de desarrolladores y usuarios colaboradores (por lo general, ubicados en distintos lugares), lo cual se traduce en una variedad de maneras de pensar que contribuye enormemente en el desarrollo y mejora del software.

- Apego a estándares de desarrollo.

  La definición y apego a estándares de desarrollo es un tarea prioritaria para el desarrollo de software libre, pues ésta facilita el trabajo colaborativo entre los desarrollos de la comunidad, y, a su vez facilita el proceso de apropiación del software por parte de los usuarios.

- Herramientas de comunicación.

  Las herramientas de comunicación al igual que los estándares de desarrollo son determinantes para el trabajo colaborativo en la construcción de aplicaciones de software. En las comunidades de desarrollo existe preferencia por el uso de herramientas de fácil uso, con lo cual se busca

promover la colaboración de desarrolladores o usuarios. Entre las herramientas que predominan se encuentran el e-mail y las listas de correo. http://www.basilv.com/psd/blog/2007/top-five-essential-practices-for-developing-software

- Plan de acción y reglas básicas construidas por la comunidad de desarrollo. Estas reglas determinan la organización en relación a las tareas que realizan los miembros de la comunidad.

2. Prácticas Características de la Ingeniería del Software

La Ingeniería de Software se define como una disciplina de la ingeniería encargada del estudio de prácticas, metodologías y herramientas para el desarrollo y mantenimiento de aplicaciones de software bajo un cronograma de entregas y unos costos estimados (Zelkovitz, 1978). Los principales objetivos de esta disciplina son:

- Mejorar la calidad de las aplicaciones de software.
- Aumentar la productividad de los desarrolladores.
- Facilitar el control del proceso de desarrollo.
- Suministrar a los desarrolladores las herramientas, prácticas y metodologías que les faciliten la construcción de aplicaciones de software que cumplan con los requerimientos de calidad establecidos por los usuarios.

La Ingeniería de Software se caracteriza por un conjunto de etapas o fases que definen todo proceso de desarrollo de software, a saber, análisis, diseño, construcción, pruebas, instalación, mantenimiento y gestión. Estas fases o etapas están constituidas por un conjunto de prácticas de desarrollo basadas en procedimientos repetibles, eficientes y efectivos, que han sido probados a través del tiempo por muchos desarrolladores de software, lo cual los ha convertido en prácticas estándares de desarrollo.

Cabe destacar que en la Ingeniería de Software existen varios modelos o procesos de desarrollo, entre los más conocidos se encuentran: Cascada, Prototipos, Espiral, Desarrollo por Etapas, Desarrollo Iterativo e Incremental, RAD (*Rapid Application Development*), Desarrollo Concurrente, Proceso Unificado, RUP (Proceso Unificado de Rational) (http://es.wikipedia.org/wiki/Ingenier %C3%ADa_de_software). Estos modelos se basan en las fases o etapas de desarrollo características de la Ingeniería de Software, pero presentan diferencias respecto al alcance de los mismos (en términos del cumplimiento de las actividades de cada fase) y a la secuencia en que se dan las fases de desarrollo en cada uno de estos modelos.

A continuación se presenta un resumen de las prácticas de desarrollo más importantes de la Ingeniería de Software, las cuales representan procedimientos fundamentales para el desarrollo de aplicaciones de software que cumplan con los requerimientos de calidad establecidos por los usuarios, y que deban ser desarrolladas bajo cronogramas de entrega y costos estimados (http://searchsoftwarequality.techtarget.com/definition/best-practice).

- Desarrollo iterativo.

Esta práctica de desarrollo se basa en la construcción de prototipos de aplicaciones de software, en donde cada prototipo representa un número específico de funcionalidades (requerimientos) de la aplicación que son desarrolladas en una iteración. De esta manera, la suma total de prototipos representa el desarrollo de todas las funcionalidades especificadas para una aplicación (http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente). Cabe destacar que en cada iteración se repite el proceso de desarrollo, es decir, se llevan a cabo las fases características de la Ingeniería de Software.

El objetivo principal del desarrollo iterativo es construir prototipos de aplicaciones con los cuales los usuarios puedan interactuar, a fin de que éstos prueben los prototipos y puedan reportar errores y sugerir cambios. Esta interacción permite a su vez que el equipo de desarrollo pueda saber, en las primeras iteraciones, si lo que ha desarrollado es lo que esperan los usuarios y si efectivamente la aplicación puede ser desarrollada en la fecha prevista (http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente).

El desarrollo iterativo se guía por la priorización de funcionalidades de la aplicación en función del valor que éstas aportan a los usuarios, así como por la priorización de los riesgos de desarrollo en función de construir en las primeras iteraciones aquellas funcionalidades que representen altos riesgos de desarrollo.

En el desarrollo iterativo no es necesario realizar una recolección completa y detallada de todos los requerimientos que debe cumplir la aplicación a desarrollar, pues al comienzo de cada iteración se recolectan de manera detallada solo los requerimientos que serán desarrollados en dicha iteración. Sin embargo, al momento de comenzar un proyecto de software bajo desarrollo iterativo se requiere conocer de manera general los requerimientos que debe cumplir la aplicación a desarrollar, con el fin de establecer el plan del proyecto.

- Administración de requerimientos.

  Esta práctica es considerada como determinante para el aseguramiento de calidad tanto a nivel del proceso de desarrollo como a nivel de las aplicaciones de software. La administración de requerimientos comprende desde la definición, especificación, revisión, asignación y control de los requerimientos de una aplicación, por lo cual es una actividad que abarca todo el proceso o ciclo de desarrollo de una aplicación de software (http://macroos0.tripod.com/n_de_requerimiento3.htm). La administración de requerimientos forma parte de la gestión de configuración del software.

  La práctica de administración de requerimientos es una de las actividades de mayor importancia dentro de la gestión de un proyecto de software, pues estos proyectos se caracterizan por una constante modificación de requerimientos y adición de nuevos requerimientos. Estas modificaciones y/o adiciones de requerimientos pueden llevar tanto a pequeños como a grandes cambios en el código de una aplicación, por lo cual al momento de decidir incluirlas o no en el desarrollo de la aplicación es importante determinar que tanto éstas afectan el desarrollo actual.

La actividad central de esta práctica es la especificación de requerimientos, la cual puede ser llevada a cabo a través de varias técnicas, entre ellas la más utilizada son los Casos de Uso (Modelado Funcional). La especificación de requerimientos constituye la explicación detallada, bajo un lenguaje natural, entendible a los usuarios, de los requerimientos que debe cumplir una aplicación de software.

La construcción de una aplicación, así como el plan de pruebas y la aplicación de éste utilizan como insumo principal la especificación de requerimientos. En tanto, dicha especificación constituye la base de todo proceso de desarrollo y en ella se fundamenta la calidad de las aplicaciones desarrolladas.

- Diseño de software.

  En esta práctica se define la arquitectura de la aplicación de software a desarrollar, para lo cual se debe describir la forma en la que se descomponen, como están organizados y como se relacionan los componentes o módulos de la aplicación (IEEEP1471-00). Cabe destacar que el diseño de software incluye también el diseño de base de datos.

  El diseño de software se considera una práctica importante tanto para la compresión del código que debe ser desarrollado, como para la compresión de éste una vez desarrollado. En el caso último, la compresión del código desarrollado, facilita la revisión de la lógica de programación para casos en los que, por ejemplo, se requiera agregar nuevos módulos, componentes o clases, o se requiera identificar errores a nivel de arquitectura de software.

  La arquitectura de una aplicación puede ser representada desde diferentes perspectivas, entre ellas las más utilizadas son: perspectiva estructural y la perspectiva de comportamiento.

  La perspectiva estructural incluye el modelado de clases, objetos, componentes, despliegue y paquetes. El modelado de clases es una de las perspectivas más utilizadas para representar la arquitectura de aplicaciones de software, para lo cual se utilizan, generalmente, los diagramas de clases.

  La perspectiva de comportamiento incluye el modelado de estados, de interacción y de actividades. Estos tipos de modelado no son tan utilizados como el modelado de clases, por lo general, se utilizan si la aplicación a desarrollar amerita un diseño más detallado que el que permite el modelado de clases. Por ejemplo, en casos en los que se requiera representar la interacción entre los elementos (métodos, funciones, etc.) dinámicos de una aplicación de software y los mensajes enviados entre ellos, se utiliza el modelado de interacción.

  En lo que respecta al diseño de base de datos el modelado más utilizado para representar los datos que se manejan en una aplicación de software y las relaciones entre éstos es el modelo de entidad-relación.

- Pruebas de software.

Las pruebas constituyen un elemento fundamental dentro del proceso de desarrollo de software, pues en base a éstas se determina si una aplicación cumple o no con los requerimientos de calidad establecidos por los usuarios. Cabe destacar que los requerimientos de calidad de una aplicación se clasifican en dos niveles, funcionales y no-funcional, por lo cual las pruebas de software se clasifican en pruebas funcionales y pruebas no-funcionales. Es importante destacar que dentro de las pruebas de software también se encuentran las pruebas unitarias, con la diferencia de que éstas no se aplican para comprobar cumplimiento de requerimientos de usuarios sino para comprobar que las unidades de software (funciones, métodos, etc.) funcionan correctamente por separado (Yagüe y Garbajosa, 2009).

La práctica de pruebas de software incluye la elaboración del plan de pruebas, su respectiva aplicación y la elaboración del reporte con los resultados de las pruebas aplicadas. Existen dos técnicas formales de pruebas: la técnica caja blanca y la técnica caja negra. La segunda técnica es la más utilizada y se basa en probar las funcionalidades de las aplicaciones desarrolladas, para lo cual los casos de prueba se basan en las diferentes entradas que puede recibir el software y sus correspondientes valores de salida (Yagüe y Garbajosa, 2009).

En muchos proyectos de desarrollo y en la Ingeniería de Software se considera que los programadores no deberían formar parte del equipo de pruebas de software, pues el hecho de que un programador codifique y realice las pruebas funcionales y no funcionales de una aplicación puede traer como consecuencia que se omitan algunos casos de prueba, por considerarse irrelevantes o por no percatarse de la necesidad de realizarlos. En este sentido, se recomienda que los probadores sean distintos de los desarrolladores, y, de ser posible, de los analistas y diseñadores de software. Con ello se busca evitar sesgos al momento de realizar las pruebas de software.

- Aseguramiento de calidad.

  Esta práctica abarca tanto el aseguramiento de calidad del software como del proceso de desarrollo de éste. Existen varios modelos para el aseguramiento de calidad, entre los más importantes se encuentran: CMMI, Norma ISO/IEC 12007, Mosca, entre otros. Los dos primeros se orientan hacia la mejora del proceso de desarrollo, mientras que el segundo se orienta al aseguramiento de calidad en el software y a la mejora continua del proceso de desarrollo.

  Los modelos para el aseguramiento de calidad del software se basan en un conjunto de métricas orientadas a la evaluación del cumplimiento de requisitos funcionales y no-funcionales de las aplicaciones de software. Los modelos de aseguramiento de calidad en el proceso de desarrollo de software se basan en un conjunto de buenas prácticas de desarrollo, y su objetivo es evaluar el proceso de desarrollo a fin determinar su calidad en función del cumplimiento o no de estas prácticas. Estas prácticas de desarrollo se basan fundamentalmente en la documentación de los subprocesos o fases que componen el proceso de desarrollo, por ejemplo, la documentación de

la especificación y administración de requerimientos, la documentación de diseño (diagramas de clases, de secuencia, etc.), la documentación de pruebas, entre otras.