Propuesta de Mejoras a la Primera Versión de la Metodología de Desarrollo de Software Libre

Fecha: 10-06-2013
Revisión: 0.1
Realizado por: Johanna Alvarez Cooz

En función de observaciones planteadas por algunos integrantes del equipo de desarrollo de la Fundación Cenditel, así como una serie de recomendaciones derivadas del Proceso de Aseguramiento de Calidad propuesto en la Fundación, se presentan a continuación un conjunto de mejoras a los procesos que componen la Metodología de Desarrollo de Software Libre. Dichas mejoras se plantean a fin de consolidar un conjunto de prácticas adecuadas a las necesidades de desarrollo de la Fundación, por lo cual se requiere estandarizar prácticas ágiles de documentación y pruebas de software destinadas a facilitar los procesos de apropiación (uso y mejora) de las aplicaciones que se desarrollen, así como a satisfacer los requerimientos de usuarios establecidos para las misma.

Mejoras requeridas en el Proceso de Conceptualización de Proyectos de Software:

- En la primera versión de la metodología de desarrollo el proceso de Conceptualización de Proyectos de Software esta compuesto de cinco (5) plantillas de documentación de los productos que se deben generar en este proceso, estas son: a) Propuesta de solución; b) Alcance del proyecto; c) Estudio de factibilidades de desarrollo, d) Posibles actores de la comunidad de desarrollo; e) Propuesta de desarrollo del proyecto. Para agilizar el proceso de Conceptualización se plantea la necesidad de generar una sola plantilla automatizada para dicho proceso, denominada Propuesta de Desarrollo, la cual contenga gran parte de la información contenida en las plantillas mencionadas anteriormente, en específico información referida a las siguientes secciones: a) necesidades/problemas, b) solución propuesta, c) alcance de la propuesta, d) arquitectura del software, e) equipo de trabajo, f) recursos/requerimientos, g) metodología de desarrollo, h) plataformas de operación y desarrollo, i) licencias de código y documentación.

  En lo respectivo a la información a presentar en la sección “alcance de la propuesta”, se plantea mostrar de forma gráfica las funciones del software (utilizando para ello diagramas de casos de uso) con las cuales podrá interactuar el usuario, a fin de tener una idea general y a la vez precisa de las tareas que el usuario podrá realizar con el software.

  Para establecer el alcance de la propuesta de desarrollo de software se requiere tener un conocimiento general de los requerimientos funcionales y de los procesos a automatizar, a fin de definir las funciones que debe cumplir el software y con la cuales interactuará el usuario. Por ello, es fundamental realizar como parte del proceso de Conceptualización actividades respectivas al proceso de Desarrollo de Aplicaciones de Software, en específico actividades de las fases de Especificación de Requerimientos y Análisis del Dominio de la Aplicación. En lo que respecta a la primera fase mencionada se requiere definir de forma general los

requerimientos funcionales del software, mientras que en el caso de la segunda fase es necesario estudiar cada proceso para determinar las entradas, los recursos, las salidas, las reglas y los actores relacionados a los mismos. Adicionalmente, se recomienda determinar la relación entre los procesos a automatizar, pues de esta relación depende la prioridad de automatización de los mismos.

- Automatización de la plantilla de "Propuesta de Desarrollo", lo cual incluye la posibilidad de poder acceder desde esta plantilla a la herramienta a utilizar para elaborar los diagramas de casos de uso.

Mejoras requeridas en el Proceso de Administración de Proyectos de Software:

- Se plantea la necesidad de eliminar la plantilla de "Descripción de la Aplicación", teniendo en consideración que la información que debe contener esta plantilla es similar a información que se registra en otras plantillas en las cuales se hace referencia a la descripción del software, como por ejemplo, en la plantilla de "Propuesta de Desarrollo". En este sentido, se considera una redundancia de información elaborar la plantilla de "Descripción de la Aplicación", para lo cual se requiere tiempo que podría ser empleado en otras actividades o prácticas relacionadas al desarrollo del proyecto de software.

- Con el objetivo de agilizar la práctica de Elaboración del Plan del Proyecto se plantean las siguientes mejoras:
    - Eliminar las plantillas de Priorización de Funcionalidades y Priorización de Riesgos, incluyendo la información solicitada en las mismas en la Plantilla del "Plan del Proyecto", ya que dicha información se requiere en esta plantilla para efectuar el cálculo de priorización de desarrollo de funcionalidades del software.
    - Automatizar la plantilla del "Plan del Proyecto", lo cual incluye la automatización del método de priorización de desarrollo de funcionalidades. Esta plantilla debe contener la siguiente información: lista de funcionalidades, priorización de funcionalidades según los usuarios y/o interesados, dependencia entre funcionalidades, lista de riesgos de desarrollo, priorización de riesgos, resultados del cálculo de priorización de desarrollo de funcionalidades y el cronograma de desarrollo por iteración.
    - Mejorar el método de priorización de desarrollo de las funcionalidades del software, con el objetivo de eliminar la subjetividad presente en el actual método, dado que en éste se utilizan factores de ponderación para las funcionalidades y los riesgos referidos al nivel de importancia de éstos para el desarrollo del proyecto, lo cual resta objetividad a dicho método.
- Con el propósito de facilitar el proceso de apropiación del software a desarrollar, así como promover el desarrollo colaborativo del mismo, se plantea la publicación de un sitio web en el que se coloque información de utilidad tanto para el uso como para la mejora del software. En este sentido, la información a publicar abarca el código fuente y la documentación asociada al mismo (manuales, especificación de requerimientos, documentación de diseño, pruebas, entre otros documentos.). Igualmente, es muy importante para facilitar el desarrollo colaborativo del

software colocar en el sitio web del proyecto información sobre medios de contactos del equipo de desarrollo del software, así como herramientas para el reporte de errores.

- En lo que respecta a la "Conformación de la Comunidad de Desarrollo", se propone eliminar el conjunto de actividades definidas en la metodología para la conformación de esta comunidad, dado que dichas actividades no se encuentran orientadas a la práctica como tal del desarrollo colaborativo, en tanto que, solo tratan asuntos referidos a la firma de acuerdos de trabajo y definición de tareas pendientes por realizar en el desarrollo de software. En su defecto, se plantea hacer énfasis en promover y facilitar el desarrollo colaborativo del software a través del uso de herramientas y mecanismos sencillos, que permitan a los usuarios e interesados en el software participar en el desarrollo del mismo, ya sea a través del reporte y/o corrección de errores o, a través del desarrollo de funcionalidades específicas y su respectiva documentación.

- En la actividad respectiva a la "Elaboración del Plan por Iteración" se considera conveniente hacer uso de herramientas que faciliten la elaboración y seguimiento de este plan. En el caso de la Fundación se utiliza la herramienta *Trac* como plataforma para la gestión del proceso de desarrollo del software, la cual cuenta con un sistema de tickets muy apropiado para la asignación de tareas. En este sentido, en lo que respecta al desarrollo de software en la Fundación, se plantea utilizar el sistema de tickets del Trac para generar el plan por iteración.
- En lo respectivo al "Seguimiento de las Tareas que Realiza el Equipo de Desarrollo" y la "Integración al Proyecto de los Aportes de los Colaboradores", se propone agrupar las mismas en una sola actividad denominada "Seguimiento y control del proyecto de software", la cual estaría compuesta por las siguientes tareas: a) reuniones periódicas de corta duración, presenciales o virtuales, entre los miembros del equipo de desarrollo para discutir asuntos del proyecto; b) seguimiento de las actividades del equipo de desarrollo; c) gestión de errores; d) seguimiento del cumplimiento de estándares.

  En la tarea de gestión de errores se plantea que el responsable de dicha tarea (Administrador del Proyecto) realice una revisión de los errores reportados (ya sea por parte del equipo de probadores (aseguramiento de calidad) o por parte de los usuarios del software) a fin de que pueda asignar la corrección de éstos a los integrantes del equipo de desarrollo, conforme a las funcionalidades que estos hayan desarrollado. Esta asignación debe ser realizada a través de alguna herramienta de registro y seguimiento de tareas, en el caso de la Fundación Cenditel se utilizará el sistema de control de tickets del *Trac* para realizar dicha asignación.

  Para facilitar la tarea de seguimiento del cumplimiento de estándares se debe indicar las herramientas que se pueden utilizar para realizar este seguimiento.

Mejoras requeridas en el Proceso de Desarrollo de Aplicaciones de Software Libre:

- Mejoras para la fase de Análisis del Dominio de la Aplicación:

  - Dado el planteamiento de incluir actividades de la fase de Análisis del Dominio de la Aplicación en el proceso de Conceptualización, en específico las respectivas a la

identificación de las reglas, los objetivos, los actores, los recursos, las entradas y las salidas de cada proceso a automatizar, así como las relaciones entre éstos, como parte de esta fase solo resta la elaboración de los diagramas o flujos de actividades que componen cada proceso y su respectiva validación con los usuarios.

- Automatización de la plantilla del diagrama de actividades, lo cual implica la posibilidad de poder acceder desde esta plantilla a la herramienta a utilizar para elaborar tales diagramas.

- Mejoras para la fase de Especificación de Requerimientos:

  - Es importante cambiar el nombre de esta fase por "Gestión de Requerimientos", pues en ella se realizan otras actividades respectivas a los requerimientos del software que involucran no solo la especificación de éstos, sino también la definición de los mismos, así como la actualización de dicha especificación. Cabe destacar la importancia que adquiere la actualización de la especificación de requerimientos para los procesos de apropiación del software, pues es común que durante el período de tiempo en que se desarrolla el software se den cambios a nivel de los requerimientos, los cuales deben quedar documentados a fin de brindar al usuario una especificación de requerimientos acorde con las funcionalidades que componen el software.
  - A fin de no coartar la creatividad del programador al codificar cada requerimiento funcional es necesario que en la especificación de requerimientos la descripción de las funcionalidades del software no se incluyan aspectos de diseño, como por ejemplo, formas de presentar o solicitar información.
  - Eliminar la actividad de validación de la especificación de requerimientos funcionales por parte de los usuarios, dado que es más fácil para el usuario validar prototipos funcionales o no funcionales del software, en los cuales puedan observar como efectivamente se vera el sistema, lo cual facilita el poder sugerir modificaciones sobre el mismo, según las tareas que el software debe realizar.
  - Incluir como parte de esta fase la validación de la especificación de los requerimientos funcionales por parte de los programadores del software, pues dicha especificación es el insumo principal que los programadores deben utilizar para codificar las funcionalidades que debe cumplir el software. Esta validación permite mejorar la especificación de requerimientos planteada por los analistas, y, a su vez agilizar el proceso de codificación de las funciones allí especificadas.
  - Automatización de la plantilla de definición de requerimientos funcionales y no funcionales.
  - Automatización de la plantilla de especificación de requerimientos funcionales, lo cual implica adicionalmente la posibilidad de poder acceder desde esta plantilla a la herramienta a utilizar para elaborar los diagramas de casos de usos.
  - A fin de poder dar un seguimiento más efectivo a los requerimientos, en términos de conocer el estado de los mismos durante el desarrollo del proyecto de software, es pertinente relacionar tanto las plantillas de definición y especificación de requerimientos, como la de los planes de pruebas funcionales y no funcionales.

- Mejoras para la fase de Análisis y Diseño:

  - Con respeto a la arquitectura del software:
    - Para definir la arquitectura de un software es necesario tener claro tanto los requerimientos funcionales como los no funcionales (restricciones y características de calidad), así como las restricciones tecnológicas existentes, pues la arquitectura seleccionada debe satisfacer estos requerimientos y su ejecución debe ser factible conforme a las capacidades con las cuales cuente el equipo de desarrollo. En este sentido, se considera pertinente incluir como una actividad principal en la fase de Análisis y Diseño la revisión y análisis de los requerimientos (funcionales y no funcionales), así como el análisis de las limitaciones tecnológicas, con la finalidad de que se converse sobre éstos como base esencial para la selección de una arquitectura apropiada. Por tanto, esta revisión y análisis debe darse entre: las personas encargadas de especificar los requerimientos, los programadores y la persona que debe diseñar la arquitectura del software.

      Cabe destacar que en la práctica de selección de la arquitectura del software se recomienda tener en cuenta además algunos aspectos como: a) la facilidad de hacer, a futuro, modificaciones o agregado de nuevas funcionalidades en el software sin que esto afecte el funcionamiento del mismo, y, sin que se requiera realizar modificaciones en todo o gran parte del código; b) estudiar varias alternativas arquitectónicas, lo cual permite poder decidir, conforme a los aspectos mencionados anteriormente, cuales de éstas satisfacen en mayor grado los requerimientos establecidos.

      En función de la importancia de los aspectos mencionados en este ítem se considera necesario incluirlos en la fase de Análisis y Diseño, a modo de recomendación para mejorar la práctica de definición de la arquitectura del software.
    - En lo que respecta a los tipos de diagramas a utilizar para representar la arquitectura del software, entre los cuales se encuentran diagramas de casos de uso, diagramas de clase, diagramas de interacción, diagramas de estado y diagramas de componentes, es importante acotar en la fase de Análisis y Diseño que el uso de los mismos dependerá del tipo de aplicación que se desarrolle y de la complejidad de la misma. Por ejemplo, en el caso de una aplicación desarrollada bajo Programación Orientada a Objeto uno de los diagramas más significativos a elaborar es el diagrama de clases, pues en el mismo se representa los elementos básicos para facilitar la fase de Codificación del Software.
    - Automatización de las plantillas para elaboración de diagramas de interacción (diagramas de secuencia, etc.), de clases, de estado y de componentes. Esta automatización implica adicionalmente la posibilidad de poder acceder desde cada una de estas plantillas a las herramientas a utilizar para elaborar los diagrama respectivos.

- Con respecto a los datos persistentes:
    - Considerando la necesidad de intercambio de datos que puede presentarse entre distintos software, es necesario incluir en la fase de Análisis y Diseño actividades respectivas a la interoperabilidad entre sistemas, entre estas actividades se encuentran: a) identificación y especificación de datos a intercambiar, b) definición de formatos de intercambio de datos.

- Mejoras para la fase de Construcción:

    - Dado que la aplicación de pruebas unitarias es una actividad de los programadores del software se requiere colocar la misma como parte de la fase de Construcción, pues en la primera versión de la metodología ésta formaba parte de la fase de Pruebas.
    - Dado las bondades que ofrecen los *framework* de desarrollo se recomienda incluir como parte de las actividades de la fase de Codificación la selección y uso de estas herramientas para facilitar y agilizar las tareas de codificación. En este sentido, es importante recomendar algunos de los *framework* mas utilizados según el lenguaje de programación.
    - A fin de facilitar la apropiación del código fuente del software desarrollado es necesario la documentación de dicho código, por lo cual es importante mencionar en la metodología algunas las herramientas que se utilizan para tal fin.
    - Es imprescindible hacer referencia en la fase de Codificación al uso de herramientas para el control de versiones, para lo cual se requiere mencionar en la metodología algunas de las herramientas más conocidas y utilizadas en esta área.
    - Para facilitar la instalación y desinstalación de software se plantea la necesidad de incluir como parte de la fase de Construcción la actividad respectiva a la automatización de los pasos requeridos para tales fines. Es importante que durante el proceso de instalación se permita al usuario suspender dicho proceso, y que se muestren a éste mensajes de información sobre el éxito o no de la instalación. De igual manera, para facilitar el uso del software se recomienda que durante el proceso de instalación se creen los iconos para el acceso al software.
    - Con respecto a la construcción de la interfaz de usuario:
      Teniendo en cuenta que el tema de la interfaz de usuario es fundamental para facilitar el uso del software, y considerando que el equipo de desarrollo de la Fundación Cenditel no cuenta con personal destinado a la práctica de diseño y desarrollo de interfaz, es pertinente que en la Metodología de Desarrollo se planteen algunas recomendaciones en relación a dicha práctica. Entre estas recomendaciones se encuentran:
      a) La interfaz de las operaciones que ejecuta el software debe mantener un estándar visual. Por ejemplo, las funciones para modificar o eliminar información deben seguir un mismo esquema de presentación en la interfaz para las distintas funciones u operaciones del software en las cuales se requieran éstas.
      b) La estructura de iconos o botones que sirven de enlace a las funciones que ejecuta el software debe ser lo más sencilla y entendible posible, es decir, se debe evitar la ejecución de varios pasos a efectuar en el software para acceder a alguna de sus

funciones.
c) La interfaz debe mantener una estandarización en relación al formato de los iconos o botones mostrados.
d) Los botones o iconos utilizados en la interfaz para acceder a las funciones u operaciones del software pueden mostrar textos en los cuales se indique que función cumple cada uno de éstos. Para ello se recomienda hacer uso de los *tool tips* .
e) Los tipos y tamaños de las letras utilizadas en las pantallas deben facilitar la visualización de los textos o frases que se presentan en la interfaz.
f) Los colores utilizados en cada pantalla deben ser contrastantes entre sí, a fin de facilitar la lectura de la información mostrada en la interfaz .
g) La interfaz debe mantener una estandarización en relación al idioma de las personas que usarán el software.

- Mejoras para la fase de Pruebas:

  - Dado que al aplicar pruebas funcionales se verifica también la integración entre los componentes o módulos del software, se considera pertinente, a fin de agilizar el proceso de desarrollo del software, eliminar de la fase de Pruebas las actividades respectivas a la elaboración y aplicación de pruebas de integración.
  - Es fundamental que se realicen pruebas de regresión al software cada vez que se agreguen nuevas funcionalidades a éste, es decir, cada vez que se culmine una iteración. En este sentido, en la metodología se debe indicar que es necesario utilizar herramientas que permitan ejecutar pruebas de regresión a partir de las pruebas funcionales que se genere en cada iteración de desarrollo. Cabe destacar que las pruebas de regresión constituyen las mismas pruebas funcionales, pero con la diferencia que las pruebas de regresión son aplicadas cada vez que se agregan nuevas funcionalidades al software, pues el objetivo de las mismas es verificar que no se hallan introducido defectos al software al agregar nuevas funciones.
    Es importante que se recomienden en la metodología algunas de herramientas para ejecutar pruebas de regresión.
  - Automatización de las plantillas para los planes de pruebas funcionales.
  - Es importante eliminar de la fase de Pruebas la actividad respectiva a la elaboración de plan de pruebas de instalación, pues las pruebas referidas consisten básicamente en llevar a cabo la instalación y desinstalación del software siguiendo los pasos descritos en el documento de instalación/desinstalación, por lo cual no se requiere elaborar un plan de pruebas como tal, dado que el objetivo de este tipo de prueba es el de verificar que el software se puede instalar y desinstalar siguiendo los pasos descritos para tales fines.
  - A fin de facilitar la ejecución de la práctica de gestión de errores se recomienda eliminar la plantilla de reporte de errores, y, en su lugar definir un esquema u herramienta a través de la cual, tanto los probadores como los usuarios del software puedan reportar directamente al Administrador del Proyecto los errores encontrados en el software. Para ello se podría automatizar un formulario de reporte de errores que este público.
  - Para agilizar el proceso de pruebas es fundamental utilizar herramientas para la

aplicación de pruebas funcionales, de regresión y no funcionales. Por ello se debe indicar en la metodología algunas de las herramientas que se podrían utilizar para tales fines.

- Agregar la actividad de "Elaboración de Manuales" como una nueva fase del proceso de Desarrollo, pues esta actividad no solo incluye la elaboración del manual de usuarios sino también la elaboración del manual de instalación. Cabe destacar la importancia de esta fase para el proceso de apropiación del software. A continuación se mencionan algunas consideraciones que deben ser indicadas en la metodología de desarrollo a fin de mejorar la práctica de elaboración de manuales:
  - En lo que respecta al manual de usuario se recomienda incluir en el mismo información como: 1) Índice. 2) Descripción general del software y de sus funciones, así como de las tareas que pueden ser ejecutadas utilizando el software. 3) Descripción de los símbolos y convenciones presentadas en la interfaz. 4) Explicación sobre el modo de uso de cada una de las funciones que componen el software, incluyendo imágenes o fotos de la interfaz que el software presenta para cada una de esta funciones. 5) Sección para preguntas frecuentes. 6) Ejemplos para ayudar en la compresión de asuntos determinados. 7) Explicación de los mensajes de error, cuando se considere necesario.
  - En relación al manual de instalación se recomienda incluir en el mismo información respectiva a: 1) Tipo de procesador, memoria RAM y dispositivos de entrada y salida de datos (Cd-Rom, micrófono, teclado, escáner, CD y/o DVD, altavoces, auriculares, tarjeta de sonido, impresoras, etc.) requeridos para colocar el software en funcionamiento, así como el tamaño del dispositivo de almacenamiento que requiere el software. 2) Sistemas operativos en los cuales funciona el software. 3) Paquetes de software requeridos para la instalación del software en los sistemas operativos indicados. 4) Información respectiva a la configuración de software en los sistemas operativos indicados. 5) Pasos requeridos para instalar y desinstalar el software en los sistemas operativos indicados. 6) Indicar si existe un procedimiento automatizado para la instalación y desinstalación del software.
  - Existen varias herramientas que se utilizan para la elaboración de manuales, por lo cual se recomienda indicar en la metodología algunas herramientas para su elaboración.

- <u>Mejoras para la fase de Liberación:</u>
  - A fin de que el software sea portable, es decir, operable en varias distribuciones, se recomienda incluir en esta fase la actividad respectiva a la realización del proceso de empaquetado del software para más de una distribución Linux.
  - La liberación del software incluye tanto la publicación de su código fuente como de la documentación (documentación generada en cada uno de los procesos de la metodología) asociada al mismo, por lo cual en esta fase se debe incluir actividades orientadas a ambos tipos de publicación, incluyendo en lo que respecta a la liberación del software la publicación de tanto de las versiones de prueba como de las versiones estables. Es importante que se indique a los usuarios el sistema o la herramienta a utilizar para reportar los errores encontrados en el software tanto a la versiones de

prueba como a las versiones estables.

- Para establecer la diferencia entre las versiones del software que se liberen es necesario utilizar un sistema de numeración de versiones que permita identificar tales diferencias entre una versión y otra. Este sistema de versiones debe permitir reflejar los cambios en el código fuente a nivel de: corrección de errores, agregado de nuevas funcionalidades y cambios fuertes en el código fuente debido modificaciones realizadas.