Aseguramiento de Calidad en el Desarrollo de Software Libre

Fundación CENDITEL

Mayo 2013

Índice general

Capítulo 1

Aseguramiento de Calidad en el Desarrollo de Software Libre

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, así como los modelos y normas para el aseguramiento de calidad en procesos y productos de software, constituyen elementos determinantes para lograr el cumplimiento de los requerimientos de calidad de las aplicaciones de software.

En el caso del ámbito del software libre son muy pocas las metodologías desarrolladas para guiar los procesos de desarrollo, así como son pocos los esfuerzos que se han llevado a cabo para desarrollar modelos de aseguramiento de calidad para este ámbito. Por lo general, en los desarrollos de software libre se utilizan métodos ágiles para la construcción del software, los cuales tiene similitudes con las metodologías ágiles como Programación Extrema[1]. En estos métodos y metodologías ágiles son muy pocas las prácticas de documentación de software que se llevan a cabo, pues lo fundamental para éstos es la construcción del código fuente. En este sentido, 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.

[1] http://www.willydev.net/descargas/Articulos/General/xplibreap.aspx

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, plantea la elaboración de un proceso para el Aseguramiento de Calidad en el Desarrollo de Software Libre, el cual permita mejorar las prácticas de desarrollo del software a fin de construir aplicaciones de software libre que cumplan estándares de calidad definidos para las mismas. Para ello el proceso propuesto hace énfasis en la mejora de las prácticas de documentación, facilitando así la apropiación de las aplicaciones que se desarrollen.

El propceso planteado consta de dos sub-procesos:

1. Evaluación del Proceso de Desarrollo de Software Libre, orientado al seguimiento del proceso de desarrollo con el objetivo de mejorar las prácticas que componen dicho proceso.

2. Evaluación de la Calidad en Aplicaciones de Software Libre, orientado a determinar la calidad de las aplicaciones en función del cumplimiento de los requerimientos de usuarios.

Los procesos anteriores requieren de un equipo de trabajo que ejecute las actividades de aseguramiento de calidad que componen los mismos, lo cual amerita la definición de una estructura organizacional y una dinámica de funcionamiento para este equipo. A continuación se definen en detalle los procesos propuestos para el Aseguramiento de Calidad en el Desarrollo de Software Libre, así como la estructura organizacional del equipo que se requiere para llevar a cabo dichos procesos.

1.1. Evaluación del Proceso de Desarrollo de Software Libre

La forma en que se lleva a cabo el proceso de desarrollo de una aplicación es determinante para la calidad de la misma, es por ello que todo proceso o modelo de aseguramiento de calidad para aplicaciones de software debe considerar la evaluación de las prácticas que componen el proceso de desarrollo. La Evaluación del Proceso de Desarrollo de Software Libre que se plantea en este modelo tiene como objetivo la mejora continua del proceso de desarrollo, razón por la cual se busca:

- Detectar oportunamente las deficiencias que se presenten en el proceso, es decir, detectar la ejecución inadecuada de prácticas de desarrollo, asegurando que éstas sean tratadas a tiempo.

- Asegurar que en el proceso se sigan los estándares establecidos respecto a la forma en la cual se deberían llevar a cabo las prácticas de desarrollo.

- Facilitar la generación de estadísticas que permitan realizar mejores estimaciones para proyectos futuros en cuanto a presupuesto, recursos y tiempo.

- Mejorar las prácticas de documentación en el desarrollo de software libre para facilitar los procesos de apropiación del software, y por tanto, facilitar procesos de estudio y mejora del software.

Existen un conjunto de normas internacionales para el área de evaluación de procesos de desarrollo de software, entre las cuales una de las más utilizadas es la norma ISO/IEC 15504. De igual forma existen varios modelos orientados a la evaluación y mejora de los procesos de desarrollo de software, entre ellos se encuentran: Capability Maturity Model Integration (CMMI), Software Process Improvement Capability Determination (SPICE), COMPETISOFT (Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica), Modelo Sistémico de Calidad (MOSCA), entre otros.

La mayoría de los modelos y las normas mencionadas plantean la evaluación de los procesos de desarrollo en función de un conjunto de preguntas orientadas a medir el rendimiento y capacidad de desarrollo de dichos procesos. El objetivo de estas preguntas, por lo general, es el determinar si en el proceso evaluado se han generado un conjunto de productos propios de un proceso de desarrollo de referencia, el cual se considera como un modelo de desarrollo a seguir. Cabe destacar que este tipo de evaluación no toma en cuenta la forma en la cual se llevan a cabo las prácticas de desarrollo, solo importa para dicha evaluación la existencia o no de los productos derivados de estas prácticas.

La evaluación de procesos de desarrollo que se propone tiene como objetivo, más que determinar el grado de rendimiento y capacidad de las organizaciones que desarrollan software, la búsqueda de la mejora continua de estos procesos. Para lograr esta mejora se plantea realizar una evaluación a las prácticas llevadas a cabo durante el desarrollo de una aplicación de software,

a fin de verificar si dichas prácticas se corresponden con buenas prácticas de desarrollo características del área de la ingeniería del software y del ámbito de desarrollo de software libre, en específico del ámbito de desarrollo de software llevado a cabo en la Fundación CENDITEL.

A continuación en las secciones 1.1.1 y 1.1.2 se presenta un resumen de las prácticas características de la ingeniería de software y del estilo de desarrollo de software libre, las cuales serán tomadas como base para la evaluación que se plantea.

1.1.1. 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 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 sólo permite que otras personas puedan utilizar el software, sino que facilita y promueve el reporte y corrección de errores del 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 una revisión mas exhaustiva del software que la que podrían hacer solo los desarrolladores del proyecto, además de facilitar las propuestas de mejoras en términos de modificaciones o desarrollo de nuevas funcionalidades.

La publicación del código, su documentación y la información relevante del proyecto se realiza, en la mayoría de los casos, a través de plataformas de desarrollo disponibles en Internet, tales como Trac, Gforge.

Comunidad de desarrollo. La conformación de una comunidad en torno al desarrollo del software permite contar con una variedad de desarrolladores y usuarios colaboradores que pueden estar ubicados geográficamente en distintos sitios, 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 desarrolladores 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. [2] .

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.

1.1.2. Prácticas Características de la Ingeniería de 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[3]. Los principales objetivos de esta disciplina son:

- Mejorar la calidad de las aplicaciones de software
- Aumentar la productividad de los desarrolladores

[2] `http://www.basilv.com/psd/blog/2007/top-five-essential-practices-for-developing-software`

- 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) [3]. 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[4].

Desarrollo iterativo. Esta práctica de desarrollo se basa en la construcción sucesiva 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[5]. 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.

[3] `http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software`
[4] `http://searchsoftwarequality.techtarget.com/definition/best-practice`
[5] `http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente`

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[6].

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. 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[7]. 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

[6] `http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente`
[7] `http://macroos0.tripod.com/n_de_requerimiento3.htm`

central de esta práctica es la eElaboración de la propuesta de desarrollospecificación 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.

Administración de cambios y defectos. La administración de cambios y defectos representa una actividad muy importante para todo desarrollo de software, dado que se requiere de una coordinación entre el equipo de desarrollo para organizar las actividades respectivas a la evaluación y asignación de cambios y defectos reportados para las aplicaciones desarrolladas.

Modelado de Software mediante el UML (Unified Modeling Language). El UML se ha convertido en un estándar para el modelado de aplicaciones de software, dado que éste permite ilustrar tanto la organización de los datos de una aplicación como el comportamiento del software a desarrollar, y, al mismo tiempo facilita la comprensión entre los desarrolladores y los usuarios.

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, la organización y la interrelación de los componentes o módulos de la aplicación (IEEEP1471-00).

El diseño de software se considera una práctica importante tanto para la comprensión del problema para desarrollar el código fuente como para la comprensión del código una vez desarrollado. En un proyecto la compresión del código fuente 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.

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-funcionales, por lo cual las pruebas de software se clasifican de la misma manera 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[1].

Probar 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 se toman en cuenta los casos de prueba con

las diferentes entradas que puede recibir el software y sus correspondientes valores esperados de salida[1].

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 además realice las pruebas funcionales y no funcionales de una aplicación puede traer como consecuencia que se omitan algunos casos de prueba, por considerarse irrelevantes. 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 el aseguramiento de calidad del software y del proceso de desarrollo de éste. El aseguramiento de calidad requiere de una constante verificación y evaluación del código y su documentación, así como también del proceso de desarrollo empleado.

La verificación consiste en una revisión de la consistencia entre los documentos de diseño del software y los requerimientos de usuarios. La evaluación respecto al código del software desarrollado consiste en determinar, en base a una serie de pruebas, si éste cumple con los requerimientos de calidad establecidos por los usuarios. La evaluación en términos del proceso de desarrollo consiste, por lo general, en una comparación entre las prácticas establecidas en un modelo de proceso de referencia y las práctica llevadas a cabo para el desarrollo de aplicaciones de software. En base a esta comparación se puede determinar deficiencias en relación a la ejecución de prácticas de desarrollo que puedan afectar la calidad del software.

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 a la evaluación del proceso de desarrollo, mientras que MOSCA se orienta al aseguramiento de calidad en el software y a la evaluación del proceso de desarrollo[2].

1.1.3. Método de Evaluación del Proceso de Desarrollo de Software Libre

En el Cuadro 1.1 se presenta el formato planteado para la evaluación de procesos de desarrollo de software libre, en el cual se describe la siguiente información:

Etapa del Proceso de Desarrollo: en este campo se indican las etapas en que está conformado el proceso de desarrollo de una aplicación de software, las cuales se corresponden con las etapas presentadas en la Metodología de Desarrollo de la Fudación CENDITEL.

Práctica: en este campo se indican las prácticas o actividades que componen cada etapa del proceso de desarrollo.

Preguntas para evaluar la práctica: en este campo se indican un conjunto de preguntas relacionadas a la ejecución de las prácticas del proceso de desarrollo a evaluar.

Opciones de Respuesta: en este campo se indican las opciones de respuesta para las preguntas plantedas en relación a cada práctica.

Responsable: en este campo se indica el(los) responsable(s) de la evaluación de cada práctica de desarrollo.

Etapa del proceso de desarrollo	**Práctica**	**Preguntas para evaluar la práctica**	**Opciones de respuesta**	**Responsable**
Conceptualización de Proyectos de Software	Elaboración de la propuesta de desarrollo	¿La propuesta de desarrollo planteada puede considerarse adecuada para el tipo de problemas o necesidades que se abordan con la misma?	Nada Poco Medianamente Ampliamente Completamente	Revisor de Conceptualización del Proyecto
		¿Se han estudiado diferentes opciones respecto a los sistemas que se pudieran desarrollar o reutilizar para solventar las necesidades y/o problemas planteados?	Si No	
		En caso de que se tengan distintas opciones de sistemas a desarrollar, ¿se ha determinado la factibilidad de la implantación de estas opciones?	Nada Poco Medianamente Ampliamente Completamente	
		¿La arquitectura general presentada para el software propuesto se basa en algún patrón de arquitectura?	Si No	
		¿El alcance del proyecto se ha definido en base a algún documento de requerimientos de usuarios o manuales de los procesos a automatizar?	Si No	
		¿El alcance del proyecto se ha especificado de manera que se pueda tener una idea clara del tamaño y complejidad del software a desarrollar?	Si No	
Administración de Proyectos de Software	Elaboración del plan del proyecto	¿Para la planificación de desarrollo del proyecto se consideró el orden de prioridad que para los usuarios tiene las funcionalidades del software?	Si No	Revisor de Conceptualización del Proyecto

Etapa del proceso de desarrollo	Práctica	Preguntas para evaluar la práctica	Opciones de respuesta	Responsable
		¿Para la planificación de desarrollo del proyecto se consideró el orden de dependencia entre las funciones o módulos del software?	Si No	
		¿Para la planificación de desarrollo del proyecto se consideró un estudio de los riesgos de desarrollo del software?	Si No	
		¿La desagregación presentada en relación a las funcionalidades del software es apropiada para determinar el tiempo de ejecución de las mismas?	Si No	
		¿Se han planificado varias iteraciones de desarrollo?	Si No	
	Definición y utilización de estándares de desarrollo	¿Se han definido estándares de desarrollo (como estándares de codificación, de interfaz, etc.)?	Nada Poco Medianamente Ampliamente Completamente	
		¿Se han utilizados estándares preestablecidos?	Si No	
		¿Se realiza seguimiento al uso de los estándares de desarrollo en el proyecto?	Si No	
		¿Se utilizan analizadores del código para verificar cumplimiento de estándares de codificación?	Si No	
Construcción de Aplicaciones de Software	Definición del dominio del software	A fin de entender la lógica secuencial entre los procesos que se requieren automatizar ¿se ha elaborado un diagrama de relación entre estos procesos, en el cual se evidencie las entradas y salidas de los mismos?	Si No	Revisor de Conceptualización del Proyecto

Etapa del proceso de desarrollo	Práctica	Preguntas para evaluar la práctica	Opciones de respuesta	Responsable
		¿En los diagramas de procesos se han especificado las reglas que regulan los procesos?	Si No	
		¿Para elaborar los diagramas de actividades que describen cada proceso se han considerado manuales de procedimientos?	Si No	
		¿Para elaborar los diagramas de actividades que describen cada proceso se han considerado las formas actuales en la cuales se llevan a cabo los procesos?	Si No	
		¿Para elaborar los diagramas de actividades que describen cada proceso se han considerado leyes que regulen la ejecución de procesos?	Si No	
		¿Se han elaborados diagramas de actividades en los cuales se propongan mejoras que faciliten la ejecución de los procesos de manera eficaz, eficiente, y efectiva?	Si No	
		¿En los diagramas de actividades se ha especificado los actores que ejecutan las mismas?	Si No	
		¿Se ha validado con los usuarios los diagramas de procesos y actividades?	Si No	
		¿Se utilizan herramientas para el modelado de procesos y actividades?	Si No	
	Definición de requerimientos de usuario	¿Los requerimientos (funcionales y no-funcionales) de usuario han sido definidos de forma que el equipo de desarrollo pueda estimar aproximadamente el esfuerzo y tiempo requerido para construir cada requerimiento?	Si No	Revisor de Requerimientos

Etapa del proceso de desarrollo	Práctica	Preguntas para evaluar la práctica	Opciones de respuesta	Responsable
		¿Los requisitos están definidos de forma clara? (Por ejemplo, un requisito poco claro o ambiguo podría ser algo como: "el software debe ser fácil de usar").	Nada Poco Medianamente Ampliamente Completamente	
	Administración de requerimientos	¿En la especificación de requerimientos se describen aspectos de diseño o de implementación?	Nada Poco Medianamente Ampliamente Completamente	Revisor de Requerimientos
		¿En la especificación de requerimientos se dan respuestas a todos los escenarios posibles?	Nada Poco Medianamente Ampliamente Completamente	
		¿En la especificación de cada requerimiento se indican los usuarios que harán uso de la funcionalidad especificada?	Nada Poco Medianamente Ampliamente Completamente	
		¿La especificación de requerimientos tiene una sintaxis correcta según la técnica y/o diagrama utilizado para tal especificación?	Si No	
		¿Se utiliza alguna herramienta para gestionar el control de cambios en la especificación de requerimientos?	Si No	
		¿Se utiliza algún tipo de técnica para la especificación de requerimientos (por ejemplo, diagramas de casos de uso)?	Si No	

Etapa del proceso de desarrollo	Práctica	Preguntas para evaluar la práctica	Opciones de respuesta	Responsable
		¿Se utilizan herramientas para elaborar la especificación de requerimientos?	Si No	
		¿Se ha validado con el equipo de desarrollo la especificación de requerimientos?	Si No	
	Definición de la arquitectura del software	¿Se han especificado claramente los requerimientos no-funcionales del software?	Si No	Revisor de Requerimientos y/o Revisor de Arquitectura del Software
		¿Se han evaluado varias alternativas de arquitectura (patrones de arquitectura)?	Si No	Revisor de Arquitectura del Software
		¿Para la evaluación de las alternativas de arquitecturas se ha tomado en cuenta los requerimientos de tipo funcional y no funcional, así como limitaciones tecnológicas existentes?	Si No	
		¿La arquitectura seleccionada permite hacer a futuro modificaciones o agregado de nuevas funcionalidades en el software, sin que esto afecte el funcionamiento del mismo?	Si No	
		¿La arquitectura seleccionada permite hacer modificaciones o agregado de nuevas funcionalidades al software, sin necesidad de realizar modificaciones en todo o gran parte del código del software?	Si No	
		¿Los diagramas utilizados para representar la arquitectura del software son los adecuados según la complejidad y alcance del software a desarrollar?	Nada Poco Medianamente Ampliamente Completamente	

Etapa del proceso de desarrollo	Práctica	Preguntas para evaluar la práctica	Opciones de respuesta	Responsable
		¿Se utilizan herramientas para representar la arquitectura del software? Como diagramas de clase, de secuencia, etc.	Si No	
	Especificación de datos persistentes	¿La especificación de los datos persistentes tiene una sintaxis correcta, según la técnica y/o diagrama (por ejemplo, diagrama entidad-relación) utilizado?	Si No	Revisor de Datos Persistentes
		En caso de utilizar base de datos relacional ¿los niveles de normalización utilizados pueden considerarse adecuados según la complejidad de la base de datos?	Nada Poco Medianamente Ampliamente Completamente	
		¿Se utilizan herramientas para especificar datos persistentes?	Si No	
		¿Se han definido formatos de intercambio de datos?	No aplica Si No	
		¿Se han especificado los datos que se van a intercambiar?	No aplica Si No	
	Codificación	¿Las pruebas unitarias son efectuadas utilizando herramientas automatizadas?	Si No	Revisor de Código
		¿Se utiliza algún framework de desarrollo?	Si No	
		¿Se ha seleccionado el framework de desarrollo según los requerimientos no-funcionales del software y según el tipo de lenguaje de desarrollo utilizado?	Si No	
		¿Se utilizan patrones de programación?	Si No	

Etapa del proceso de desarrollo	Práctica	Preguntas para evaluar la práctica	Opciones de respuesta	Responsable
		¿Se utilizan herramientas para generar la documentación del código fuente?	Si No	
		¿Se utilizan herramientas para el control de versiones del código?	Si No	
	Elaboración y aplicación de pruebas funcionales/no-funcionales	¿El plan de pruebas funcionales se ha elaborado en base a alguna técnica o método de prueba (ejemplo caja negra)?	Si No	Probador de Requerimientos Funcionales/No-Funionales
		¿Los casos de prueba para evaluar cada función del software abarcan todos los escenarios que se pueden presentar al ejecutar cada una de estas funciones?	Nada Poco Medianamente Ampliamente Completamente	
		¿Se cuenta con herramientas para aplicar pruebas de regresión?	Si No	
		¿El plan de pruebas no funcionales se ha elaborado en base a alguna técnica o método de prueba (ejemplo caja negra)?	Si No	
		¿Se cuenta con herramientas para aplicar pruebas no funcionales?	Si No	
		¿Las pruebas funcionales son aplicadas por personas distintas a los desarrolladores del software?	Si No	
		¿Las pruebas no funcionales son aplicadas por personas distintas a los desarrolladores?	Si No	
		¿Se utilizan herramientas para la gestión de los errores reportados en las pruebas del software?	Si No	

Etapa del proceso de desarrollo	**Práctica**	**Preguntas para evaluar la práctica**	**Opciones de respuesta**	**Responsable**
	Liberación de versiones	¿El proyecto de desarrollo de software cuenta con un sitio web donde se publique información respectiva al proyecto (código, documentación, etc.)?	Si No	Revisor de Conceptualización del Proyecto
		¿Se publica la documentación (diagramas de procesos, especificación de requerimientos, etc.) asociada a las versiones del software que son liberadas?	Nada Poco Medianamente Ampliamente Completamente	
		¿Se utiliza algún sistema de numeración para las versiones liberadas del software?	Si No	
		¿El sistema de numeración utilizado para la liberación de versiones del software permite identificar cambios en el código fuente por bugs, por funcionalidades y por cambios fuertes en el mismo?	Nada Poco Medianamente Ampliamente Completamente	
		¿Existe liberación de versiones de pruebas (betas) del software?	Si No	
	Interacción	¿Se realizan reuniones presenciales o virtuales periódicas entre los miembros del equipo de desarrollo para discutir asuntos del proyecto ?	Nada Poco Medianamente Ampliamente Completamente	Cualquier miembro del Equipo de Aseguramiento de Calidad
		¿Se utilizan herramientas para facilitar la interacción entre los miembros del equipo de desarrollo del software?	Si No	
		¿Se cuenta con herramientas que permitan la interacción entre usuarios y el equipo de desarrollo del software?	Si No	

Cuadro 1.1: Evaluación del Proceso de Desarrollo de Software Libre

Es importante resaltar que en los casos en que debido al tipo de software que se desarrolla no se amerite llevar a cabo algunas de las prácticas indicadas en el Cuadro 1.1, éstas no deben ser contempladas en la evaluación que se realice a dicho proceso.

La evaluación propuesta para cada práctica de desarrollo debe ser realizada al mismo tiempo en que la práctica es ejecutada, ello con la finalidad de que se pueda corregir de inmediato las disconformidades que se observen durante la ejecución de la misma, evitando así la ejecución incorrecta de prácticas que pueda conllevar a la disminución de la calidad de la aplicación de software que se desarrolla.

1.2. Evaluación de la Calidad en Aplicaciones de Software Libre

Este proceso tiene como objetivo evaluar en las aplicaciones de software libre el cumplimiento de las características o requerimientos de calidad establecidos para las mismas. La calidad de un software se evalúa en términos de sus requerimientos funcionales y no-funcionales. En relación a los requerimientos funcionales se evalúa la capacidad operativa del software, es decir, se evalúan las funcionalidades que este ofrece en relación a las necesidades de los usuarios. En los requerimientos no-funcionales se evalúan cualidades o propiedades emergentes del software como la fiabilidad, la mantenibilidad, los tiempos de respuesta, la capacidad de almacenamiento, entre otras, que indican al usuario el comportamiento del software bajo ciertas condiciones del ambiente donde éste será puesto en producción.

La evaluación de la calidad de una aplicación de software se realiza en función de métricas. En los modelos y en las normas de aseguramiento de calidad de software la métrica se define como una medida del grado en el cual una aplicación cumple con un requerimiento de usuario, también se define como una medida del grado en el cual un proceso de desarrollo cumple con los estándares establecidos para llevar a cabo el desarrollo de una aplicación.

Las métricas constituyen el elemento fundamental en el cual se basan las normas y los modelos de aseguramiento de calidad de software. En las normas y en los modelos, las métricas están orientadas a medir atributos o características de calidad como:

- Funcionalidad: estas métricas permiten evaluar en que medida el soft-

ware satisface los requerimientos funcionales prescritos de acuerdo a las necesidades de los usuarios.

- Fiabilidad: estas métricas se utilizan para determinar la madurez y rendimiento de un software.
- Usabilidad: estas métricas permiten medir el esfuerzo que deben realizar los usuarios para aprender a usar el software.
- Eficiencia: estas métricas son utilizadas para evaluar el rendimiento del software en función de la cantidad de recursos utilizados por éste.
- Mantenibilidad: estas métricas permiten medir el nivel de esfuerzo requerido para modificar el software.
- Portabilidad: estas métricas se utilizan para determinar el esfuerzo necesario para transferir el software de un entorno de sistema hardware y/o software a otro entorno diferente.

Existen varios modelos y normas de calidad de software como: Modelo de MCCALL, MOSCA y Normas ISO para aplicaciones de software, entre ellas ISO/IEC 12207, ISO/IEC 9126-1, ISO/IEC 9126-2, ISO/IEC 9126-3, ISO/IEC 9126-4, ISO/IEC 9241, ISO/IEC 12119 e ISO/IEC 14598. Las Normas ISO para software constituyen el estándar más conocido y utilizado en la mayoría de los modelos y métodos de evaluación de software.

En relación a la norma ISO/IEC 9126-4 es importante destacar que a pesar de que la misma constituye uno de los estándares internacionales más utilizados en modelos y/o métodos para evaluar calidad del software, existen algunas dificultades para la aplicación del proceso de evaluación presentado en dicha norma. Algunas de estas dificultades o complicaciones se pueden apreciar, por ejemplo, en observaciones derivadas de un ejercicio de evaluación de software realizado en la Fundación CENDITEL, en el cual se valido la primera versión del Modelo de Aseguramiento de Calidad en el Desarrollo de Software Libre, basado éste en la norma ISO/IEC 9126-4. Estas observaciones se muestran a continuación:

- Para aplicar varios de los métodos de evaluación propuestos en la referida norma, respectivos a métricas de usabilidad, se requiere contar con un equipo numeroso de probadores de software. En este sentido, para proyectos de software que no cuentan con personal encargado de las

tareas de aseguramiento de calidad, o que cuentan con equipos pequeños, tal como en el caso del ejercicio de evaluación mencionado, aplicar dichos métodos de evaluación tiende a ser complicado.

- Los métodos de evaluación de algunas métricas de usabilidad presentados en la referida norma son de carácter subjetivo, pues dependen de la consideración o puntos de vista de quienes realizan las evaluaciones.

Para solventar dificultades como las mencionadas en los ítem anteriores han sido definidas un conjunto de normas que facilitan y dan objetividad al proceso de evaluación de las características de calidad planteadas en la norma ISO/IEC 9126-4. Entre estas normas se encuentran las ISO/IEC 14598-1 y 14598-5, éstas se han utilizado en algunos métodos y/o modelos de evaluación de software. Uno de los modelos en los cuales se puede apreciar el uso de las normas ISO/IEC 14598-1 y 14598-5, para evaluar las características de calidad indicadas en la norma ISO/IEC 9126-4, es el Método de Avaliação de Qualidade de Produto de Software (MEDE-PROS)[4], desarrollado en Brasil.

El proceso de evaluación presentado en MEDE-PROS puede ser utilizado para evaluar cualquier tipo de software, más sin embargo, en éste no se considera la evaluación de características de calidad referidas a mantenibilidad. El método de evaluación propuesto en MEDE-PROS se basa en un conjunto de cuestionarios fundamentados en las normas ISO/IEC 14598-1 y 14598-5, donde las preguntas que se plantean son formuladas en función de verificar si el software evaluado cumple o no con las características de calidad presentadas en la norma ISO/IEC 9126-4. Las posibilidades de respuesta a las preguntas planteadas en MEDE-PROS se encuentran definidas en un rango de opciones, lo cual no permite que la evaluación dependan de la subjetividad de quien realiza la evaluación.

Es importante destacar que al aplicar el proceso de evaluación de MEDE-PROS se generan una especie de informes en los cuales se reporta al equipo de desarrollo del software evaluado el cumplimiento o no de las características o requisitos de calidad. En el caso del no cumplimiento de dichas características se explica a detalle en los informes respectivos cuáles aspectos o elementos del software deben revisarse para mejorar la calidad del mismo. Con estos informes se busca generar alertas que permitan al equipo de desarrollo del software mejorar sus prácticas de desarrollo, a fin de cumplir con la características de calidad requeridas para el software. En este sentido, el método de evaluación presentado en MEDE-PROS va mas allá de la sola evaluación del software.

Teniendo en consideración los aspectos mencionados en los párrafos anteriores, así como la experiencia obtenida durante el ejercicio de evaluación de una aplicación de software libre, realizado en la Fundación CENDITEL, se plantea a continuación un método de evaluación de software inspirado en el método MEDE-PROS, en el cual las características de calidad a evaluar incluyen tanto características indicadas en la norma ISO/IEC 9126-4, así como otras características particulares al área de desarrollo de software libre, las cuales se consideran determinantes para asegurar la calidad de este tipo de aplicaciones.

1.2.1. Método de Evaluación de Calidad en Aplicaciones de Software Libre

Con este método de evaluación se busca evaluar un conjunto de aspectos o componentes de las aplicaciones de software libre que permiten medir la calidad de las mismas no solo en términos de características funcionales y de usabilidad, sino también haciendo énfasis en la evaluación de características asociadas a las facilidades de modificación de estas aplicaciones, y a las posibilidades de operabilidad de éstas en diferentes entornos de hardware y software. Al hacer énfasis en la evaluación de las dos últimas características mencionadas se persigue facilitar la modificación y adecuación del software, así como también promover el uso de éste en diferentes sistemas operativos y bajo distintos entornos de hardware, lo cual favorece la aplicabilidad de dos de las libertades fundamentales de la filosofía del software libre (a saber, uso y modificación del software). Adicionalmente, este método de evaluación promueve la mejora continua de las prácticas de desarrollo de software, en específico la mejora continua de las prácticas de documentación, pues de éstas depende, en gran medida, las facilidades de uso y modificación del software libre.

El método de evaluación que se plantea se basa en el proceso de evaluación presentado en MEDE-PROS. Cabe destacar que a diferencia de MEDE-PROS en el método que se propone se ha considerado la evaluación de algunos otros componentes o elementos del software, que son de significativa importancia para las aplicaciones de software libre, tales como: liberación o publicación de versiones y documentación de requerimientos.

Los componentes del software que se evaluan se asocian a características de calidad, tales como funcionabilidad, eficiencia, fiabilidad, portabilidad,

mantenibilidad y usabilidad. Estos componentes se presentan a continuación:

- Documentación de requerimientos: componente referido a la documentación de especificación de requerimientos funcionales del software.
- Documentación de usuarios: componente referido a los documentos impresos o en línea donde se explica a los usuarios el uso del software.
- Documentación del código fuente: componentes referido a las descripciones o especificaciones que se realizan en las funciones, clases o métodos que componen el código fuente, donde se indica información respectiva a las mismas.
- Documentación de instalación y desinstalación: componente referido a los documentos donde se especifican las instrucciones para instalar y desinstalar el software.
- Pruebas de software: componente referido a la verificación del cumplimiento de requerimientos funcionales y no funcionales.
- Interfaz de usuarios: componente referido a las facilidades de usabilidad del software.
- Empaquetado: componente que indica la posibilidad de que el software pueda ser utilizado en varias distribuciones.
- Publicación del software: componente referido a la publicación de las versiones del software, lo cual implica la liberación del código y la documentación asociada al mismo.

Al igual que en el método MEDE-PROS cada uno de los componentes indicados en los ítem anteriores son evaluados en función de un conjunto de preguntas que permiten verificar en que medida el software evaluado cumple las características de calidad asociadas a cada componente.

En el Cuadro 1.2 se presenta el método de evaluación de calidad en aplicacicones de software libre.

Componente	**Métricas asociadas al componente**	**Preguntas para evaluar el componente**	**Opciones de respuesta**	**Responsable**
Documentación de requerimientos	Funcionalidad, Mantenibilidad	¿Que proporción de cambios en el software, desde la iteración anterior, se encuentran documentadas en la especificación de requerimientos?	Proporción	Revisor de Requerimientos
Documentación de usuarios	Usabilidad	¿Que proporción de cambios en el software, desde la iteración anterior, se encuentran especificados en la documentación?	Proporción	Revisor de Manuales de Software
		¿En la documentación (impresa o en línea) se indica el nombre del software?	Si No	
		¿En la documentación (impresa o en línea) se indica la versión y la fecha de creación de la software?	Si No	
		¿En la documentación están identificadas las tareas que pueden ser ejecutadas utilizando el software?	Si No	
		¿La documentación presenta, a través de textos, imágenes o fotos, si la interfaz con el usuario se realiza a través de: líneas de comando, menús, ventanas, teclas de funciones, teclas de acceso directo, barra de botones, otros?	Si No	
		¿La documentación presenta un texto introductorio o de presentación sobre el software?	Si No	
		¿La documentación presenta una descripción general del software y de sus funciones?	Si No	
		¿En la documentación el tamaño de la letra facilita la lectura?	Si No	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿La documentación está organizada para facilitar la comprensión por parte del usuario?	Nada Poco Medianamente Ampliamente Completamente	
		¿La documentación permite una fácil identificación de las funciones?	Si No	
		¿La documentación contiene un tema dedicado a presentar al usuario los símbolos y convenciones presentadas en la interfaz?	Si No	
		¿La documentación utiliza recursos para resaltar la información pertinente. Por ejemplo: negrita, cursiva, las palabras en letras mayúsculas, números, texto sombreado?	Nada Poco Medianamente Ampliamente Completamente	
		¿En la documentación (impresa o en línea) se encontró evidencia de información incorrecta? Por ejemplo: unidades de medida, abreviaciones, notaciones, entre otras.	Si No	
		¿La documentación es clara y precisa?	Si No	
		¿La documentación presenta errores de gramática?	Si No	
		¿La documentación presenta errores ortográficos?	Si No	
		¿En la documentación se utiliza términos y explicaciones considerando el tipo de usuario que hará uso del software?	Si No	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿En la documentación se explica los mensajes de error, cuando sea necesario?	Nada Poco Medianamente Ampliamente Completamente	
		¿La documentación disponible es compatible con la versión del software?	Si No	
		¿La documentación presenta una sección para preguntas frecuentes?	Si No	
		¿La documentación presenta ejemplos para ayudar en la compresión de un asunto determinado?	Nada Poco Medianamente Ampliamente Completamente	
		¿Los ejemplos en la documentación son claros y precisos?	Si No	
		¿La documentación presenta índice general?	Si No	
		¿Los títulos de los capítulos de la documentación son los mismo que aparecen en el índice?	Si No	
		¿El usuario puede encontrar determinada información a través de la enumeración de páginas?	Si No	
		¿Utiliza recursos de hipertexto a través de link (manuales en línea)?	Nada Poco Medianamente Ampliamente Completamente	

Componente	**Métricas asociadas al componente**	**Preguntas para evaluar el componente**	**Opciones de respuesta**	**Responsable**
	Funcionalidad	¿En la documentación se observa alguna contradicción o inconsistencia en relación a la información que aparece en la interfaz del software?	Si No	
	Mantenibilidad	¿Qué proporción de funcionalidades del software están descritas en la documentación?	Proporción	
Documentación del código fuente	Mantenibilidad	¿Se encuentra documentado el código desarrollado?	Nada Poco Medianamente Ampliamente Completamente	Revisor de Código
		¿Qué proporción de funcionalidades del software están descritas en la documentación?	Nada Poco Medianamente Ampliamente Completamente	
Pruebas funcionales	Fiabilidad	¿El número de fallas aumenta de iteración en iteración?	Si No	Probador de Requerimientos Funcionales
		¿Cual es la proporción de fallas eliminadas después de la pruebas funcionales?	Proporción	
		¿Qué proporción de campos que requieren validación se están chequeando?	Proporción	
		¿Está el software preparado para la accesibilidad de usuarios con discapacidad física?	Nada Poco Medianamente Ampliamente Completamente	
		¿Qué proporción de casos de usos especificados se encuentran implementados?	Proporción	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿Las funciones del software implementadas atienden de forma completa los requerimientos de usuarios?	Si No	
		¿Las funciones del software satisfacen de forma completa las necesidades o tareas que se deben realizar?	Si No	
		¿El software cuenta con mecanismos que permitan hacer auditorías?	Si No	
		¿Se guarda una bitácora de las operaciones que se realizan dentro del software?	Si No	
		¿El software tiene la capacidad de controlar el número de intentos para introducir contraseñas?	Si No	
	Portabilidad	¿El software funciona en más de un sistema operativo, sin que el usuario perciba diferencias de funcionamiento?	Si No	
		¿Está el software preparado para la accesibilidad de usuarios con discapacidad física?	Nada Poco Medianamente Ampliamente Completamente	
Pruebas no funcionales	Eficiencia	¿Los tiempos de respuesta del software al ejecutar sus funciones satisfacen los tiempos de respuesta esperados?	Nada Poco Medianamente Ampliamente Completamente	Probador de Requerimientos no Funcionales
		¿El manejo de los recursos computacionales es eficiente en relación a las especificaciones no funcionales?	Si No	

Componente	**Métricas asociadas al componente**	**Preguntas para evaluar el componente**	**Opciones de respuesta**	**Responsable**
Interfaz de usuario	Usabilidad	¿Se mantiene un estándar visual en la interfaz de las operaciones del software?	Si No	Revisor de Interfaz
		¿Resulta fácil encontrar las funcionalidades en el software?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz está organizada en grupos de acuerdo a una forma lógica comprendida por el usuario?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz hace uso de identificadores que representa claramente su significado. Ejemplo: títulos, iconos, entre otros?	Si No	
		¿La interfaz informa a los usuarios sobre la función de un botón, menú, icono o ventana de diálogo al posicionar el cursor del ratón sobre el?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz permite deshabilitar algunos diálogos y presentaciones iniciales?	Si No	
		¿Las pantallas presentan una distribución uniforme de su contenido, teniendo en cuenta los espacios disponibles?	Nada Poco Medianamente Ampliamente Completamente	
		¿Las pantallas tienen áreas de selección de elementos de menú?	Si No	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿Las pantallas utilizan los tipos y tamaños de letra para facilitar la visualización de los campos de entrada de datos y sus formatos?	Nada Poco Medianamente Ampliamente Completamente	
		¿Las pantallas tienen colores contrastantes, por lo que es fácil de leer?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz presenta errores gramaticales?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz presenta errores ortográficos?	Nada Poco Medianamente Ampliamente Completamente	
		¿En la interfaz se destacan las palabras en otro idioma. Por ejemplo, entre comillas, negritas, cursivas, etc?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz mantiene una estandarización en relación al idioma de las personas que usarán el software?	Nada Poco Medianamente Ampliamente Completamente	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿La interfaz mantiene una estandarización en relación a la configuración de las ventanas?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz mantiene una estandarización en relación al formato de los iconos?	Nada Poco Medianamente Ampliamente Completamente	
		¿La interfaz mantiene una estandarización en relación a la posición de un determinado botón que ejecuta una misma función en ventanas de diálogo distintas (Ejemplo: botón “Cancelar”, “Imprimir”, “Aceptar”, entre otros)?	Nada Poco Medianamente Ampliamente Completamente	
Documentación de instalación/desinstalación	Usabilidad	¿Está especificado el tipo de procesador necesario para colocar el software en uso?	Si No	Revisor de Manuales de Software
		¿Está especificada la memoria RAM necesaria para colocar el software en uso?	Si No	
		¿Están especificados los dispositivos de entrada de datos necesarios para colocar el software en uso? Por ejemplo: Cd-Rom, micrófono, teclado, escáner, etc	Si No	
		¿Están especificados los dispositivos de salida de datos necesarios para colocar el software en uso? Por ejemplo: CD y/o DVD, altavoces, auriculares, tarjeta de sonido, impresoras, plotters, etc.	Nada Poco Medianamente Ampliamente Completamente	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿Se especifica el tamaño del dispositivo de almacenamiento requerido para colocar el software en funcionamiento?	Si No	
		¿Se especifican las tarjetas de expansión necesaria para poner el software en uso. Por ejemplo, tarjeta aceleradora de gráficos.?	Si No	
		¿Se indican los sistemas operativos en los cuales funciona el software?	Si No	
		¿Se especifican los paquetes de software requeridos para la instalación del software en los sistemas operativos indicados?	Nada Poco Medianamente Ampliamente Completamente	
		¿Se especifica la información requerida para la configuración de software en los sistemas operativos indicados?	Nada Poco Medianamente Ampliamente Completamente	
		¿Se orientan paso a paso como ejecutar la instalación en los sistemas operativos indicados?	Nada Poco Medianamente Ampliamente Completamente	
		¿Están claros los pasos de instalación que se deben realizar?	Nada Poco Medianamente Ampliamente Completamente	
		¿Se ofrece la información necesaria para instalar el software?	Si No	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿Se presentan las instrucciones para realizar la desinstalación del software?	Si No	
		¿Las instrucciones de desinstalación guían paso a paso cómo llevar a cabo las acciones de desinstalación?	Si No	
Pruebas de instalación/desinstalación	Usabilidad	¿Hay niveles de instalación de acuerdo a las necesidades del usuario. Por ejemplo: instalación básica o estándar, instalación personalizada?	Si No	Probador de Requerimientos no Funcionales
		¿Se muestran mensajes informando el progreso de la instalación?	Si No	
		¿Permite definir el subdirectorio en el cual se instalará el software?	Si No	
		¿Durante la instalación se crean automáticamente los subdirectorios necesarios?	Si No	
		¿Durante la instalación se copian los archivos necesarios automáticamente?	Si No	
		¿Durante la instalación se crean iconos para el acceso al software?	Si No	
		¿Durante la instalación se proporciona una función que permite detener el proceso de instalación?	Si No	
		¿Se muestra mensaje para informar sobre el éxito o no de la instalación?	Si No	
		¿Una vez completado el proceso de instalación, se puede ejecutar el software?	Si No	
		¿El usuario necesita interactuar con el equipo de desarrollo para instalar el software?	Si No	

Componente	Métricas asociadas al componente	Preguntas para evaluar el componente	Opciones de respuesta	Responsable
		¿El tiempo que toma la instalación del software a un usuario es igual al tiempo promedio de instalación definido por el equipo de desarrollo?	Nada Poco Medianamente Ampliamente Completamente	
		¿El software presenta un procedimiento automatizado o manual para realizar la desinstalación?	Si No	
		¿Si tiene un procedimiento para la desinstalación automatizada: muestra mensajes que indican el progreso de la tarea?	Si No	
		¿Si tiene un procedimiento para la desinstalación automatizada: borra los archivos copiados en la instalación del software?	Si No	
		¿Si tiene un procedimiento para la desinstalación automatizada: elimina los subdirectorios creados durante la instalación del software?	Si No	
		¿Si tiene un procedimiento para la desinstalación automatizada: avisa al usuario sobre la necesidad de reiniciar el sistema, después de completar la desinstalación?	Si No	
		¿Si tiene un procedimiento para la desinstalación automatizada: elimina los iconos asociados, en su caso?	Si No	
		¿Si tiene un procedimiento para la desinstalación automatizada: muestra el mensaje final sobre el éxito o fracaso de la desinstalación se ha completado?	Si No	

Componente	**Métricas asociadas al componente**	**Preguntas para evaluar el componente**	**Opciones de respuesta**	**Responsable**
Empaquetado de liberación	Usabilidad	¿Está el software empaquetado por lo menos para una distribución linux?	Si No	Probador de Requerimientos no Funcionales
		¿Está el software empaquetado para más de una distribución linux?	Si No	
Publicación del software	Usabilidad, Mantenibilidad	¿Las versiones del código fuente se encuentran publicadas en un sitio web?	Si No	Revisor de Conceptualización del Proyecto
		¿Las versiones de la documentación asociada al software (por ejemplo, diagramas de procesos, especificación de requerimientos, arquitectura, etc.) se encuentran publicadas en un sitio web?	Nada Poco Medianamente Ampliamente Completamente	

Cuadro 1.2: Evaluación de Calidad en Aplicaciones de Software Libre

Es importante mencionar que dado el tipo de software desarrollado puede que no se requiera la evaluación de algunos de los componentes planteados en el Cuadro 1.2, por lo cual en la evaluación del software no se debe considerar dichos componentes.

1.3. Equipo de Aseguramiento de Calidad en el Desarrollo de Software Libre

A fin de llevar a cabo el proceso de Aseguramiento de Calidad en el Desarrollo de Software Libre, es necesario definir una estructura organizacional destinada a realizar las actividades de evaluación propuestas en el mismo. Antes de pasar a describir la estructura mencionada se presentan algunas consideraciones que deben ser tomadas en cuenta para la conformación de dicha estructura:

- Las personas encargadas de llevar a cabo el proceso de aseguramineto de calidad deben tener un amplio conocimiento y experiencia en el desarrollo de software, así como en la implementación de metodologías para el desarrollo de software.
- El tamaño de las organizaciones que desarrollan software es otro elemento a considerar, pues las prácticas de aseguramiento de calidad requieren personal con capacidad tanto en el área de programación como en las áreas de análisis, diseño, pruebas y documentación. En las orgnizaciones desarrolladoras de software que estan conformadas por pequeños grupos de trabajo, casi siempre constituidas solo por programadores, la ejecución de prácticas de aseguramiento de calidad tienden a ser muy poca, pues no se cuenta con el personal requerido para muchas de estas pŕacticas. Teniendo este elemento en consideración se plantea una estructura para el equipo de aseguramiento de calidad fundamentada en el concepto de rol, lo cual facilita el hecho de que una persona puede cumplir varios roles a la vez en los distintos procesos de evaluación, siempre y cuando posea las competencias requeridas para llevar a cabo los mismos.

La estructura del equipo de aseguramiento de calidad se plantea en función de los siguientes roles, los cuales serán descritos en las secciones siguientes:

- Coordinador del Equipo de Aseguramiento de Calidad.
- Revisor de la Conceptualización del Proyecto (RCP).
- Revisor de Requerimientos (RR).
- Revisor de la Arquitectura de Software (RAS).
- Revisor de Datos Persistentes (RDP).
- Revisor de Código (RC).
- Probador de Requerimientos Funcionales (PRF).
- Probador de Requerimientos no Funcionales (PRNF).
- Revisor de Manuales de Software (RMS).

1.3.1. Coordinador del Equipo de Aseguramiento de Calidad (CEAC)

La persona que ejerza este rol debe tener conocimiento y experiencia en el desarrollo de software, pues este rol tiene como responsabilidad realizar el seguimiento y control de las actividades que lleva a cabo el equipo de aseguramiento de calidad, en relación a la evaluación de las prácticas de desarrollo y al producto de software. Entre sus principales actividades se encuentran:

- Asignar responsables con capacidades y tiempo suficiente para desempeñar los demás roles del equipo de aseguramiento de calidad.
- Realizar el monitoreo y el seguimiento de las actividades planificadas por el equipo de aseguramiento de calidad.
- Llevar un histórico en el que se registren las disconformidades observadas en las prácticas de desarrollo y en los productos de software, adicionando a ello el registro de las recomendaciones que planteen los evaluadores en función de las disconformidades reportadas. El objetivo de este registro es que la organización desarrolladora de software cuente con una especie de base de conocimiento donde queden registradas buenas prácticas de trabajo, que hayan contribuido a mejorar la calidad

de los productos desarrollados. Esta especie de base de conocimientos permitirá mejorar futuros procesos de desarrollo.

1.3.2. Revisor de la Conceptualización del Proyecto (RCP)

La persona que desempeñe este rol debe tener capacidades para la administración de proyectos de software, lo cual implica tener conocimiento y experiencia en lo referente al proceso de desarrollo como al proceso de aseguramiento de calidad del software. Entre las principales actividades que debe realizar el RCP, se mencionan las siguientes:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en el Cuadro 1.1.
- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.
- Evaluar el componente del software asociado a este rol conforme se propone en Cuadro 1.2.
- Elaborar el reporte de evalución del componente asociado a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o para componentes de software en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.3.3. Revisor de Requerimientos (RR)

La persona que desempeñe este rol deben tener conocimiento y experiencia en relación al área de administración de requerimientos.

Entre las principales actividades que debe realizar el RR se encuentran:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en la Cuadro 1.1.
- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.

- Evaluar el componente de software asociados a este rol conforme se propone en Cuadro 1.2.
- Elaborar el reporte de evalución de los componentes asociados a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o componentes en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.3.4. Revisor de la Arquitectura de Software (RAS)

La persona que ejerce este rol deben tener capacidades en relación al planteamiento y diseño de los componentes o módulos que integran el software, la manera cómo éstos deben estar organizados y cómo se deben relacionar entre sí. Para ello debe tener capacidades en el modelado de clases, de objetos, de componentes, de despliegue, de paquetes, de estados, de interacción y de actividades, entre otros.

Entre las principales actividades que realiza el RAS se mencionan las siguientes:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en la Cuadro 1.1.
- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.
- Evaluar los componentes de software asociados a este rol conforme se propone en el Cuadro 1.2.
- Elaborar el reporte de evalución de los componentes asociados a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o componentes de software en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.3.5. Revisor de Datos Persistentes (RDP)

Para la evaluación del diseño de los datos persistentes de una aplicación de software se requiere de personal con conocimiento y experiencia en el modelado de base de datos. Entre las principales actividades que debe realizar el RDP se mencionan las siguientes:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en la Cuadro 1.1.
- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.
- Evaluar los componentes de software asociados a este rol conforme se propone en el Cuadro 1.2.
- Elaborar el reporte de evalución de los componentes asociados a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o componentes de software en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.3.6. Revisor de Código (RC)

El RC se encarga de la evaluación del código fuente de las aplicaciones de software, por lo cual la persona que lleve a cabo este rol debe tener capacidad y experiencia en el área de programación.

Entre las principales actividades que debe realizar el RC se mencionan las siguientes:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en el Cuadro 1.1.
- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.
- Evaluar los componentes de software asociados a este rol conforme se propone en el Cuadro 1.2.

- Elaborar el reporte de evalución de los componentes asociados a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o componentes de software en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.3.7. Probador de Requerimientos Funcionales (PRF)

El PRF debe tener conocimiento y experiencia tanto en la formulación como en la ejecución de planes de pruebas funcionales.

Entre las principales actividades que debe llevar a cabo un PRF se encuentran:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en el Cuadro 1.1.
- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.
- Evaluar los componentes de software asociados a este rol conforme se propone en el Cuadro 1.2.
- Elaborar el reporte de evalución de los componentes asociados a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o componentes de software en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.3.8. Probador de Requerimientos no Funcionales (PRNF)

El PRNF debe tener conocimiento y experiencia tanto en la formulación como en la ejecución de planes de pruebas no funcionales. Entre las principales actividades que debe realizar el PRNF se encuentran:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en el Cuadro 1.1.

- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.
- Evaluar los componentes de software asociados a este rol conforme se propone en el Cuadro 1.2.
- Elaborar el reporte de evalución de los componentes asociados a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o componentes de software en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.3.9. Revisor de Manuales de Software (RMS)

El RMS deben tener capacidades y experiencias en la elaboración de manuales de usarios, de instalación/desinstalación y administración del software.

Entre las principales actividades que debe realizar el RMS se encuentran:

- Evaluar las prácticas de desarrollo asociadas a este rol conforme al formato de evaluación propuesto en el Cuadro 1.1.
- Elaborar el reporte de evalución de las practicas asociadas a este rol, y emitir este reporte al equipo de desarrollo del software.
- Evaluar los componentes de software asociados a este rol conforme se propone en el Cuadro 1.2.
- Elaborar el reporte de evalución de los componentes asociados a este rol, y emitir este reporte al equipo de desarrollo del software.
- Repetir la evaluación para prácticas de desarrollo y/o componentes de software en los cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

1.4. Generación de Reportes de Aseguramiento de Calidad en el Desarrollo de Software Libre

El objetivo de este proceso es la generación de reportes que permitan informar a los equipos de desarrollo de las aplicaciones de software evaluadas las disconformidades observadas tanto en el proceso de desarrollo como en las aplicaciones, así como algunos otros aspectos que se consideren pertinentes de reportar. De esta manera, los equipos de desarrollo podrán realizar las modificaciones necesarias para mejorar las prácticas de desarrollo y/o la calidad de las aplicaciones.

Las disconformidades que se pueden detectar al evaluar un proceso de desarrollo constituyen discrepancias que se observan entre la manera en que una práctica es llevada a cabo y la forma en que se considera correcta (buenas prácticas de desarrollo) ejecutar la misma, a fin de alcanzar los mejores resultados de esta ejecución. Las disconformidades que se pueden observar al evaluar un software constituyen incumplimientos de características de calidad en el software, las cuales se encuentran asociadas a requerimientos de usuario, así como a aspectos particulares que facilitan el uso y mantenibilidad del mismo.

En la evaluación del proceso de desarrollo se debe considerar como disconformidad todas aquellas respuestas a preguntas planteadas para evaluar prácticas que se encuentren entre las siguientes opciones: “No”, “Poco” y “Medianamente”. Estas opciones reflejan que la ejecución de la práctica evaluada dista de una ejecución correcta de la misma, lo cual incide en la calidad de la aplicación de software que se desarrolla.

No deberán ser consideradas como disconformidades aquellas respuestas a preguntas plateadas para evaluar prácticas de desarrollo que se correspondan con la opción “Ampliamente”, pues ella indica que la práctica evaluada es ejecuta de manera muy cercana a la forma en la cual se considera correcta. Sin embargo, en este caso, con la finalidad de promover las buenas prácticas de desarrollo, el evaluador debe resaltar la importancia de continuar mejorando la práctica evaluada con miras a lograr una ejecución correcta de la misma.

En la evaluación de aplicaciones de software se debe considerar como disconformidad todas aquellas respuestas a preguntas planteadas para evaluar componentes de software que se encuentre entre las siguientes opciones: “No”, “Poco”, “Medianamente” y “Proporción”, para esta última opción se conside-

raran solo como disconformidades aquellas proporciones que tengan valores menores a cero coma setenta y cinco (0,75). Estas opciones reflejan que la aplicación de software no cumple de manera satisfactoria las características de calidad que son requeridas en la aplicación para asegurar el cumplimiento de requisitos funcionales y no funcionales, así como para facilitar el uso y modificación de la misma.

No deberán ser consideradas como disconformidades aquellas respuestas a preguntas plateadas para evaluar componentes de software que se encuentren entre las opciones "Ampliamente" y "Proporción", en específico proporciones que tengan valores mayores o iguales a cero como setenta y cinco (0,75), pues estas opciones indican que los componentes evaluados cumple en gran parte con las características de calidad definidas para los mismos. Sin embargo, en estos casos, el evaluador debe resaltar la importancia de mejorar la calidad de los componentes evaluados, con el objetivo de desarrollar aplicaciones que cumplan de manera completamente satisfactoria las características de calidad definidas para éstas.

Bibliografía

[1] J. Garbajosa A. Yagüe. Las pruebas en metodologías Ágiles y convencionales: Papeles diferentes. In *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, volume 3, España, 2009. Universidad Politécnica de Madrid (UPM).

[2] A. Grimán L. Mendoza, M. Pérez. Prototipo de modelo sistémico de calidad (MOSCA) del software. *Revista Computación y Sistemas de la Universidad Autónoma del Estado de México*, 8(3):196–221, 2005.

[3] A. Shaw y J. Gannon M. Zelkovitz. *Principles of Software Engineering and Design*. Prentice Hall, 1979.

[4] A. Cervigni Guerra y R. Thienne Colombo. *Tecnologia da Informação: Qualidade de Produto de Software*. Fundación Biblioteca Nacional, 2006. INPI 820166243.