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

Fundación CENDITEL

12 de enero de 2012

Índice general

Capítulo 1

Modelo para el 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 cons-

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

tituye la base para realizar cualquier estudio de éste con fines de entender su funcionamiento o mejorarlo.

La Fundación Cenditel considerando lo planteado en los párrafos anteriores, y, en base a sus necesidades como Centro de Desarrollo de Tecnologías Libres, entre ellas desarrollo de software, se plantea la elaboración de un Modelo para el Aseguramiento de Calidad en el Desarrollo de Software Libre, el cual permita mejorar las prácticas de desarrollo del software, haciendo énfasis en la mejora de las prácticas de documentación, facilitando así la apropiación de las aplicaciones desarrolladas.

El modelo planteado consta de tres procesos:

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

3. Generación de Reportes de Aseguramiento de Calidad, el cual tiene como finalidad principal reportar a los equipos de desarrollo las disconformidades observadas en las prácticas de desarrollo y en el producto de software.

Los procesos anteriores requieren de un equipo de trabajo que ejecute las actividades de aseguramiento de calidad que componen los procesos, lo cual amerita de 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 modelo y 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, 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.
- Generar 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 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. Adicionalmente a los modelos mencionados existen las normas internacionales ISO para el desarrollo de software, entre ellas la norma ISO/IEC 15504 como la más utilizada en lo relacionado a la evaluación de procesos de desarrollo.

La mayoría de los modelos y las normas mencionadas plantean la evaluación de los procesos de desarrollo en función de un conjunto de métricas o indicadores orientados a medir el rendimiento y capacidad de desarrollo de dichos procesos, a fin de determinar el nivel de madurez de las organizaciones que llevan a cabo los mismos. Esta evaluación se base en una comparación entre las prácticas genéricas definidas para un proceso de referencia y las prácticas genéricas del proceso que se quiere evaluar.

En el caso del modelo propuesto se plantea una evaluación de procesos de desarrollo que, más que determinar el grado de madurez y capacidad de las organizaciones que desarrollan software, busca la mejora continua de estos procesos. Para realizar esta evaluación se puede utilizar el mismo tipo de evaluación planteada en modelos como CMMI y COMPETISOFT, en los cuales se evalúa un proceso determinado en función de su comparación con

un proceso de desarrollo de referencia. Al utilizar este tipo de evaluación se puede determinar qué prácticas del proceso de desarrollo evaluado no se están llevando a cabo conforme se plantea en las prácticas que componen el proceso de referencia. De esta manera, se buscaría la mejora continua del proceso evaluado a través de la realización de buenas prácticas de desarrollo.

Para el caso del modelo propuesto se plantea un proceso de desarrollo de referencia en el que se combinan prácticas características de la ingeniería de software con prácticas características del desarrollo de software libre. De esta combinación resulta un proceso híbrido que responde a la necesidad de mejorar las prácticas de documentación que llevan a cabo muchos equipos de desarrollo de software libre, pues en muchos de estos procesos de desarrollo se carece de buenas práctica de documentación que faciliten la apropiación de los productos desarrollados.

A continuación en las secciones 1.1.1 y 1.1.2 se presenta un resumen de las prácticas características tanto de la ingeniería de software como del estilo de desarrollo de software libre, que serán consideradas para conformar el proceso de referencia que se utilizará en el modelo para evaluar los procesos de desarrollo.

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.

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

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
- 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

[3] `http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software`

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.

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

[4] `http://searchsoftwarequality.techtarget.com/definition/best-practice`
[5] `http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente`
[6] `http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente`

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 central de esta práctica es la especificación de requerimientos, la cual puede ser llevada a cabo a través de varias técnicas, entre ellas la más utilizada son los Casos de Uso (Modelado Funcional). La especificación de requerimientos constituye la explicación detallada, bajo un lenguaje natural, entendible a los usuarios, de los requerimientos que debe cumplir una aplicación de software. La construcción de una aplicación, así como el plan de pruebas y la aplicación de éste utilizan como insumo principal la especificación de requerimientos. En tanto, dicha especificación constituye la base de todo proceso de desarrollo y en ella se fundamenta la calidad de las aplicaciones desarrolladas.

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.

[7] `http://macroos0.tripod.com/n_de_requerimiento3.htm`

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. Evaluación de las Prácticas que componen el Proceso de Desarrollo de Software Libre

Tal como se planteó en 1.1 se propone la evaluación de un proceso de desarrollo de software en base a una comparación entre un proceso de desarrollo de referencia, constituido en este caso por el conjunto de buenas prácticas indicadas en las secciones 1.1.1 y 1.1.2, y un proceso de desarrollo a evaluar. Esta comparación se realiza a través de una serie de preguntas, basadas en las prácticas del proceso de referencia, a través de las cuales se pretende detectar disconformidades en la realización de las prácticas del proceso de desarrollo a evaluar.

En el Cuadro 1.2 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. En este caso el proceso de desarrollo de referencia que se utiliza para la evaluación está constituido por tres etapas: Conceptualización de Proyectos de Software, Administración de Proyectos de Software y Construcción de Aplicaciones de Software.

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

Preguntas relacionadas a la ejecución de las tareas que componen la práctica: en este campo se indican preguntas en relación a la ejecución de las prácticas del proceso de desarrollo evaluado, las cuales son planteadas conforme a la ejecución del proceso de desarrollo que se utiliza como referencia en la evaluación.

Evaluación/Tarea: en este campo se indica la evaluación que se le asigna a cada una de las tareas que componen una práctica. Esta evaluación debe ser asignada conforme a la siguiente escala de valores presenta en el Cuadro 1.1.

Responsable de la Evaluación: en este campo se indica el(los) responsable(s) de la evaluación de cada una de las tareas que componen las prácticas de desarrollo.

Cuadro 1.1: Escala de valores para evaluar las tareas que componen las prácticas del proceso de desarrollo

Valor Cualitativo	**Valor Cuantitativo**
No	0
Parcialmente	0,25
Medianamente	0,50
Ampliamente	0,75
Completamente	1

Etapa del proceso de desarrollo	**Práctica**	**Preguntas relacionadas con la ejecución de las tareas que componen la práctica**	**Evaluación/Tarea**	**Responsable**
Conceptualización de Proyectos de Software	Elaboración del documento de la propuesta de desarrollo	¿En el documento se han identificado los problemas y/o necesidades a las que debe responder el software que se plantea desarrollar?		Revisor de Conceptualización del Proyecto
		¿En el documento se define la aplicación de software a desarrollar?		
		¿La aplicación de software planteada en el documento es adecuada para abordar los problemas y/o necesidades planteadas por los usuarios?		
		¿En el documento se define el alcance del proyecto en términos de los módulos o componentes del software que se deben desarrollar en función de los requerimientos de usuarios?		
		¿En el documento se presenta a grandes rasgos la arquitectura del software?		
		¿En el documento se establece el equipo de desarrollo del proyecto?		
		¿En el documento se definen los recursos y/o requerimientos necesarios para llevar a cabo el proyecto?		
		¿En el documento se define la metodología de desarrollo a utilizar?		
		¿En el documento se definen las plataformas de operación y desarrollo de la aplicación de software?		
		¿En el documento se establecen las licencias libres a utilizar tanto para el código como para su documentación?		

Etapa del proceso de desarrollo	**Práctica**	**Preguntas relacionadas con la ejecución de las tareas que componen la práctica**	**Evaluación/Tarea**	**Responsable**
	Publicación de documentos generados en la etapa de Conceptualización de Proyectos de Software.	¿Se han publicado todos los documentos generados para la etapa de Conceptualización de Proyectos de Software?		
Administración de Proyectos de Software	Elaboración del documento de priorización de funcionalidades del software	¿En el documento se han identificado las funcionalidades generales del software en función de los módulos o componentes de éste?		Revisor de Conceptualización del Proyecto
		¿En el documento se ha planteado el orden de prioridad entre funcionalidades conforme a las necesidades de los usuarios?		
		¿En el documento se ha indicado el orden de dependencia entre funcionalidades?		
	Elaboración del documento de estudio de riesgos de desarrollo	¿En el documento se han descrito los riesgos de desarrollo de la aplicación de software?		
		¿En el documento se ha descrito el impacto de los riesgos de desarrollo?		
		¿En el documento se han asociado las funcionalidades indicadas para el software con los riesgos de desarrollo definidos?		
		¿En el documento se ha establecido la prioridad para abordar cada uno de los riesgos?		
		¿En el documento se han definido acciones preventivas y correctivas para los riesgos?		
	Elaboración del documento del plan del proyecto	¿En el documento se ha establecido la prioridad de desarrollo para las funcionalidades del software, en base a la prioridad de funcionalidades establecida por los usuarios y en base al orden de dependencia entre funcionalidades?		

Etapa del proceso de desarrollo	Práctica	Preguntas relacionadas con la ejecución de las tareas que componen la práctica	Evaluación/Tarea	Responsable
		¿En el documento se ha definido el cronograma del proyecto en base a iteraciones de desarrollo, indicando en cada iteración el grupo de funcionalidades a desarrollar en cada una de ellas?		
	Elaboración del documento de estándares de desarrollo	¿En el documento se han definido los estándares de desarrollo que se utilizarán en el proyecto?		
	Elaboración del documento del plan por iteración	¿En el documento se han indicado las tareas de desarrollo correspondientes a las funcionalidades planteadas para la iteración actual, así como las fechas de inicio y culminación de estas tareas y los responsables de las mismas?		
	Instalación y puesta en uso de las herramientas de comunicación y trabajo colaborativo	¿Se han instalado y puesto en funcionamiento herramientas de comunicación entre los integrantes del proyecto?		
		¿En el documento se ha definido el cronograma del proyecto en base a iteraciones de desarrollo, indicando en cada iteración el grupo de funcionalidades a desarrollar en cada una de ellas?		
	Elaboración del documento de normas o reglas para facilitar el trabajo colaborativo en el desarrollo del software	¿Se ha elaborado el documento de las normas o reglas para facilitar el desarrollo colaborativo del software?		

Etapa del proceso de desarrollo	**Práctica**	**Preguntas relacionadas con la ejecución de las tareas que componen la práctica**	**Evaluación/Tarea**	**Responsable**
	Publicación de todos los documentos generados en la etapa de Administración de Proyectos de Software	¿Se han publicado todos los documentos generados en la etapa de Administración de Proyectos de Software?		
Construcción de Aplicaciones de Software: Fase de Administración de Requerimientos	Elaboración del documento de especificación de requerimientos para la iteración actual	¿En el documento se han especificado en términos gráficos y textuales los casos de uso correspondientes a las funcionalidades a desarrollar en la iteración actual?		Revisor de requerimientos
		¿En el documento se han especificado los requerimientos no funcionales del software?		
		¿En la descripción textual de los casos de uso que se plantea en el documento se presenta el flujo básico y el flujo alternativo, así como los requisitos o condiciones especiales para cada caso de uso?		
		¿La redacción de los casos de uso planteados en el documento es clara y consistente conforme a las funcionalidades establecidas para la iteración actual?		
		¿Los gráficos de los casos de uso presentados en el documento tienen una sintaxis correcta?		
	Validación de la especificación de requerimientos correspondientes a la iteración actual	¿Ha sido validado entre programadores y analistas de sistemas el documento de especificación de requerimientos?		
		¿Los usuarios de la aplicación de software han validado el documento de especificación de requerimientos?		

Etapa del proceso de desarrollo	**Práctica**	**Preguntas relacionadas con la ejecución de las tareas que componen la práctica**	**Evaluación/Tarea**	**Responsable**
	Gestión de control de requerimientos en la iteración actual	¿Se ha estudiado el impacto sobre la versión actual del software en caso de incluir, cambiar y/o eliminar requerimientos?		
		¿Se han definido estrategias requeridas para incluir, eliminar y/o cambiar requerimientos en la versión actual del software?		
		¿Se ha actualizado el documento de especificación de requerimientos en base a los cambios, eliminación o inclusión de requerimientos aprobados en la iteración actual?		
	Publicación de documentos generados en la fase de Administración de Requerimientos	¿Se han publicado todos los documentos generados en esta fase de la etapa de Construcción de Aplicaciones de Software?		
Construcción de Aplicaciones de Software: Fase de Diseño	Elaboración del documento de especificación de la arquitectura del software	¿En el documento se refleja el modelado de todas las unidades funcionales que deben componer el software?. Este modelado se realiza en función de diagramas requeridos, como diagramas de clase, de componentes, de secuencia, etc.		Revisor de Arquitectura de Software
		¿Los diagramas planteados en el documento presentan una sintaxis correcta?		
		¿En los diagramas planteados en el documento se indican adecuadamente las relaciones entre los elementos que componen estos diagramas?		
		¿Ha sido validado entre programadores y analistas de sistemas la arquitectura del software presentada en el documento?		

Etapa del proceso de desarrollo	Práctica	Preguntas relacionadas con la ejecución de las tareas que componen la práctica	Evaluación/Tarea	Responsable
	Actualización del documento de la arquitectura del software en la iteración actual	¿Se ha actualizado el documento la arquitectura del software conforme a cambios, inclusiones o eliminaciones de requerimientos planteadas en la especificación de requerimientos correspondiente a la iteración actual?		
	Elaboración del documento de especificación de los datos persistentes de la aplicación de software	¿En el documento de modelado de la base de datos se han especificado los datos persistentes de la aplicación de software?. Para el modelado de la base de datos se pueden utilizar herramientas como los diagramas de entidad-relación, entre otros. Cabe destacar que no todas las herramientas para especificación de datos se aplica a cualquier aplicación, pues éstas varían dependiendo del tipo de aplicación de software.		Revisor de Datos Persistentes
		¿Ha sido validado entre los programadores y los analistas de sistemas el documento de modelado de los datos persistentes?		
	Actualización del documento de especificación de los datos persistentes de la aplicación de software en la iteración actual	¿Se ha actualizado el documento de modelado de la base de datos conforme a cambios, inclusiones o eliminaciones de datos persistentes realizados en al iteración actual?		
	Publicación de documentos generados en la fase de Diseño	¿Se han publicado todos los documentos generados en esta fase de la etapa de Construcción de Aplicaciones de Software?		Revisor de Arquitectura de Software y Revisor de Datos Persistentes

Etapa del proceso de desarrollo	**Práctica**	**Preguntas relacionadas con la ejecución de las tareas que componen la práctica**	**Evaluación/Tarea**	**Responsable**
Construcción de Aplicaciones de Software: Fase de Construcción	Verificación del uso de los estándares de desarrollo en el código desarrollado en la iteración actual	¿Se han cumplido los estándares de desarrollo en el código construido?		Revisor de código
		¿Se ha documentado el código desarrollado?		
	Publicación de documentos generados en la fase de Construcción	¿Se han publicado todos los documentos generados en esta fase de la etapa de Desarrollo de Aplicaciones de Software?		
Construcción de Aplicaciones de Software: Fase de Pruebas	Elaboración del documento del plan de pruebas funcionales para la iteración actual	¿El documento es elaborado por probadores?		Revisor de Pruebas Funcionales
		¿En el documento se ha definido el ambiente de prueba y los recursos necesarios para realizar las pruebas funcionales?		
		¿En los casos de prueba presentados en el documento se identifican los datos de entrada, los datos de salida y los datos esperados en la versión actual del software?		
		¿En el documento se han definido casos de prueba para todos los casos de uso descritos en la especificación de requerimientos correspondiente a la iteración actual?		
		¿La redacción utilizada en los casos de prueba presentados en el documento, para la iteración actual, es clara y entendible?		
	Aplicación del plan de pruebas funcionales correspondiente a la iteración actual	¿Se ha aplicado el plan de pruebas correspondiente a la iteración actual?		

Etapa del proceso de desarrollo	Práctica	Preguntas relacionadas con la ejecución de las tareas que componen la práctica	Evaluación/Tarea	Responsable
		¿Se ha suministrado un reporte al equipo de desarrolladores para que conozcan los errores encontrados en las pruebas realizadas a la versión actual del software, a fin de su respectiva corrección?		
	Corrección de errores reportados como resultado de las pruebas funcionales correspondiente a la iteración actual	¿Sea han corregido los errores reportados en el plan de pruebas aplicado a la versión actual del software?		
		¿Se han aplicado nuevamente los casos de prueba en los que se reportaron errores para la versión actual del software, a fin de confirmar la corrección de estos errores?		
		¿Después de la corrección de los errores reportados en la versión actual del software se han aplicado nuevamente casos de prueba, distintos a los casos de prueba en los que se reportaron errores, que son necesarios para verificar que la corrección de los errores reportados no introdujo efectos no deseados en la versión actual de software?		
	Elaboración del documento del plan de pruebas no funcionales correspondiente a la iteración actual	¿Se ha definido el ambiente de prueba y los recursos necesarios para realizar las pruebas no funcionales?		Revisor de Pruebas No Funcionales

Etapa del proceso de desarrollo	Práctica	Preguntas relacionadas con la ejecución de las tareas que componen la práctica	Evaluación/Tarea	Responsable
		¿En los casos de prueba presentados en el documento se han identificado las variables de interés a medir en la versión actual del software, así como las variables de entrada con las cuales se probará dicha versión para medir la variables de interés?. Entre las variables de interés para medir en el software se encuentran: cantidad de memoria utilizada, tiempos de respuesta, entre otros		
		¿Se han definido en el documento casos de prueba para cubrir todos los requerimientos no funcionales establecidos?		
		¿La redacción utilizada en los casos de prueba es clara y entendible?		
	Aplicación del plan de pruebas no funcionales correspondiente a la iteración actual	¿Se ha aplicado el plan de pruebas?		
	Exposición de resultados de las pruebas no funcionales correspondientes a la iteración actual	¿Se ha elaborado un informe donde se indique, como resultado de las pruebas aplicadas, el rendimiento de la versión actual del software en función de la cantidad de recursos utilizados por éste?		
	Elaboración de los manuales de la aplicación de software	¿Se han elaborado los manuales del software conforme a los estándares de documentación establecidos?		Revisor de Documentación

Etapa del proceso de desarrollo	**Práctica**	**Preguntas relacionadas con la ejecución de las tareas que componen la práctica**	**Evaluación/Tarea**	**Responsable**
	Publicación de documentos generados en la fase de Pruebas	¿Se han publicado todos los documentos generados en esta fase de la etapa de Construcción de Aplicaciones de Software?		Revisor de Pruebas Funcionales, Revisor de Pruebas no Funcionales y Revisor de Documentación

Cuadro 1.2: Evaluación del Proceso de Desarrollo de Software Libre

La evaluación propuesta para cada práctica de desarrollo debe ser realizada al momento de culminar cada práctica, con la finalidad de que se pueda corregir de inmediato las disconformidades reportadas, evitando así la acumulación de disconformidades en las prácticas que pueda conllevar a la disminución de la calidad de la aplicación de software que se desarrolla. Los reportes que se deben emitir, tanto para la evaluación de las prácticas como para el reporte de disconformidades observadas en las mismas se especifican en la sección 2.4.

1.2. Evaluación de la Calidad en Aplicaciones de Software Libre

Con este proceso se tiene como objetivo determinar si las aplicaciones de software cumplen con los requerimientos de calidad establecidos por los usuarios. 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 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 la aplicación de software.

Las métricas pueden ser directas o indirectas. Las métricas directas son aquellas que no requieren de un cálculo que involucra otras métricas[4], por ejemplo, una métrica directa sería “número de enlaces rotos en una página web”. Las métricas indirectas son aquellas que se obtienen a partir de un cálculo que involucra otras métricas[4], por ejemplo, una métrica indirecta sería “porcentaje de funciones de la aplicación de software son descritas en los manuales del software”.

Las métricas constituyen el elemento fundamental en el cual se basan los modelos y las normas de calidad de software. En éstos modelos y en las normas, las métricas están orientadas a medir atributos como funcionalidad, portabilidad, mantenibilidad, fiabilidad, eficiencia y usabilidad del software. Existen varios modelos y normas de calidad de software como: Modelo de MCCALL, MOSCA y Normas ISO para aplicaciones de software (entre ellas ISO 12207, ISO 9126-1, ISO 9126-2, ISO 9126-3, ISO 9126-4, ISO 9241, ISO 12119 e ISO 14598). Las Normas ISO para software constituyen el estándar más conocido y utilizado en la mayoría de los modelos de calidad de software.

Considerando que las normas ISO/IEC para software constituyen un estándar internacional, utilizado en varios modelos para evaluar calidad del software, como por ejemplo MOSCA (Mendoza, Pérez y Griman, 2005), se plantea el proceso de Evaluación de la Calidad de las Aplicaciones de Software en base a métricas definidas en la norma ISO/IEC 9126-3 (Métricas Internas de la Calidad del Producto de Software), así como en base a algunas listas de chequeo para evaluar la calidad técnica de la documentación del software.

El proceso de Evaluación de la Calidad en Aplicaciones de Software que se plantea tiene como propósito generar alertas respecto a disconformidades que se puedan presentar en el software, en relación a los requerimientos de calidad indicados por los usuarios, los cuales se evaluarán en función de las métricas de calidad que se requieran para el mismo.

1.2.1. Métricas para evaluar calidad en Aplicaciones de Software Libre

A continuación se presentan las métricas descritas en la norma ISO/IEC 9126-3, las cuales serán utilizadas en el modelo propuesto para determinar la calidad en aplicaciones de software libre:

- Métricas de funcionalidad
- Métricas de fiabilidad
- Métricas de usabilidad
- Métricas de eficiencia
- Métricas de mantenibilidad

- Métricas de portabilidad

Cada una de estas métricas se dividen en otras submétricas, las cuales a su vez se subdividen en otras métricas más específicas, conformando así un conjunto considerable de métricas. Este conjunto de métricas obedece a que se han definido métricas para medir el grado de cumplimiento de requerimientos funcionales y el grado de cumplimiento de requerimientos no funcionales para distintos tipos de aplicaciones de software. Por esta razón, es fundamental a la hora de evaluar la calidad de un software el determinar si las métricas existentes son aplicables al software de interés, dado los requerimientos funcionales y no funcionales establecidos para el mismo. Por ejemplo, para evaluar la calidad de algunos software para juegos no se requieren métricas de interoperabilidad (métrica de funcionalidad), dado que este tipo de aplicaciones, por lo general, no requieren de interacción con otras aplicaciones.

Para determinar la calidad de cualquier tipo de aplicación de software, ya sea desarrollada bajo enfoque libre o propietario, se requiere como mínimo utilizar métricas de funcionalidad y de usabilidad, pues en función de ellas se puede medir el grado de cumplimiento de los requerimientos funcionales establecidos para la aplicación, así como la facilidad de uso de ésta. En el caso de la evaluación de calidad para aplicaciones en software libre se requiere como mínimo utilizar las mismas métricas recomendadas para las aplicaciones de software propietario, pero añadiendo las métricas de mantenibilidad, dado que es de fundamental importancia para la modificación de un software, pues miden el esfuerzo requerido para implementar una modificación especifica en el software.

El proceso de medición de las métricas indicadas para determinar la calidad de aplicaciones desarrolladas en software libre se plantea conforme al formato presentado en el Cuadro 1.3, el cual está inspirado en el formato utilizado en la norma ISO/IEC 9126-3.

Nombre de la sub-métrica:								
Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**

Cuadro 1.3: Formato de medición para métricas de calidad de software

En el Cuadro 1.3 se describe:

Nombre de la submétrica: indica el tipo de submétrica a la que corresponden las métricas específicas a describir.

Métrica específica: indica el nombre de la métrica específica que será medida.

Propósito: corresponde al objetivo de la métrica, es decir, lo que se quiere medir.

Método de medición: forma en que se mide una métrica directa.

Método de cálculo: es un algoritmo o cálculo que se realiza para combinar dos o más métricas directas o indirectas, a fin de obtener una métrica indirecta.

Fuente de medición: corresponde a la fuente de donde se obtienen los datos medidos.

Responsables: persona o rol encargado de realizar la medición de la métrica.

Nota: corresponde a observaciones que son consideradas importantes en relación al proceso de medición que se plantea en el Cuadro 1.3.

1.2.2. Métricas de Funcionalidad

Las métricas de Funcionalidad se utilizan para determinar si el software satisface los requerimientos funcionales prescritos de acuerdo a las necesidades de los usuarios. Este tipo de métricas se subdividen en:

Métricas de adecuación: indican el conjunto de atributos necesarios para evaluar explícitamente funciones para las tareas preestablecidas y para determinar su adecuación en la realización de esas tareas.

Métricas de exactitud: atributos para evaluar la capacidad del software para lograr resultados correctos o aceptables.

Métricas de interoperabilidad: atributos para evaluar la capacidad de interacción del software con otro software

Métricas de seguridad: atributos para evaluar la capacidad del software de evitar accesos ilícitos a las funciones y/o datos de éste.

Métricas de conformidad de la funcionalidad: atributos para evaluar la capacidad del software para cumplir con elementos tales como normas, convenciones o regulaciones de funcionalidad establecidos por la organización que usará el software.

Cada una de estas sub-métricas se subdividen a su vez en otras métricas específicas, en función de las cuales se miden los atributos de funcionalidad de un software. Estas submétricas y la forma en que se miden se describen en el Cuadro 1.4.

Métricas de adecuación								
Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Adecuación funcional	¿Qué tan adecuadas son las funciones implementadas?	Contar el número de funciones implementadas en las que se detectó problemas para realizar tareas especificadas y comparar con las funciones implementadas en el software	$X = 1 - A/B$ A= Número de funciones en las que se detectó problemas en la evaluación. B= Número de funciones implementadas en el software	A se obtiene del reporte de pruebas funcionales. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mejor adecuación funcional del software		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Integridad en la implementación funcional	¿Qué tan completa es la implementación funcional?	Contar el número de funciones omitidas en el software con respecto al número de funciones descritas en las especificaciones de los requerimientos	$X = 1 - A/B$ A= Número de funciones omitidas en el software con respecto al número de funciones descritas en las especificaciones de los requerimientos. B= Número de funciones descritas en las especificaciones de los requerimientos	A se obtiene del reporte de pruebas funcionales B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mejor integridad en la implementación funcional del software.		Revisor de Pruebas Funcionales y Revisor de Requerimientos	La entrada al proceso de medición es la especificación de requerimientos actualizada. Cualquier cambio identificado durante el ciclo de vida debe ser aplicado a la especificación de requerimientos antes de usarla en el proceso de medición

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Estabilidad de la especificación funcional	¿Qué tan estable es la especificación de requerimientos funcionales durante el ciclo de vida de desarrollo?	Contar el número de funciones añadidas, modificadas, o eliminadas durante la fase del ciclo de vida de desarrollo, y comparar con el número de funciones descritas en las especificaciones de los requerimientos funcionales	$X = 1 - A/B$ A= Número de funciones cambiadas durante la fase del ciclo de vida de desarrollo B= Número de funciones descritas en las especificaciones de los requerimientos.	A y B se obtienen de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor estabilidad en la especificación de requerimientos funcionales		Revisor de Requerimientos	
Métricas de exactitud								
Precisión de cálculos	¿Qué tan completos son los requerimientos de precisión implementados en los cálculos que realiza el software?	Contar el número de funciones que han implementado los requerimientos de precisión de cálculo, y comparar con el número de funciones para las cuales se necesita implementar requerimientos de precisión de cálculos específicos	$X = A/B$ A= Número de funciones en las que se han implementado requerimientos de precisión específicos B= Número de funciones para las cuales se necesita implementar requerimientos de precisión de cálculos específicos	A se obtiene del código fuente y/o del diseño. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican una mayor precisión de cálculo del software		Revisor de Código y/o Revisor de Arquitectura del Software, Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Adecuación de los tipos de datos	¿Qué tan adecuada es la implementación de los tipos de datos del software?	Contar el número de datos que cumplen con los tipos de datos especificados para éstos y comparar con el número total de datos especificados	$X = A/B$ A= Número de datos que cumplen con los tipos de datos especificados B= Número total de datos especificados	A se obtiene del código fuente y/o del diseño B se obtiene del modelo de especificación de datos persistentes	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican una implementación más adecuada de los tipos de datos.		Revisor de Código y/o Revisor de Arquitectura del Software, Revisor de Requerimientos	
Precisión	¿Qué tan adecuada es la implementación del rango de valores que pueden tomar los datos?	Contar el número de datos que cumplen con los rangos de valores especificados para éstos y comparar con el número total de datos que requieren rangos de valores específicos	$X = A/B$ A= Número de datos que cumplen con los rangos de valores especificados para éstos B= Número de datos que requieren rangos de valores específicos	A se obtiene del código fuente B se obtiene del modelo de especificación de datos persistentes y/o de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican una mayor precisión de calculo del software en relación a los rangos de valores que pueden tomar los datos		Revisor de Código, Revisor de Base de Datos y/o Revisor de Requerimientos	
Métricas de interoperabilidad								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad del software para intercambiar datos	¿Qué tan adecuada es la implementación de los formatos de datos de interfaz?	Contar el número de formatos de datos de interfaz que se implementaron adecuadamente de acuerdo a lo especificado, y comparar con el número de formatos de datos de interfaz que deben ser intercambiados según las especificaciones	$X = A/B$ A= Número de formatos de datos de interfaces que se implementaron de acuerdo a lo especificado B= Número de formatos de datos de interfaces que deben ser intercambiados según las especificaciones	A se obtiene del código fuente B se obtiene del modelo de especificación de datos persistentes y/o de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mejor capacidad del software para intercambiar datos		Revisor de Código, Revisor de Base de Datos y/o Revisor de Requerimientos	
Consistencia de la interfaz	¿Qué tan adecuada es la implementación de los protocolos de interfaz?	Contar el número de protocolos de interfaz que han sido implementados según la especificaciones, y comparar con el número de protocolos de interfaz que deben ser implementados según las especificaciones	$X = A/B$ A= Número de protocolos de interfaz que han sido implementado según la especificaciones B= Número de protocolos de interfaz que deben ser implementados según las especificaciones	A se obtiene del código fuente B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor consistencia de la interfaz		Revisor de Código, Revisor de Requerimientos	
Métricas de seguridad								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad de auditar accesos	¿Qué tan auditables son los accesos?	Contar el número de tipos de accesos que están siendo registrados correctamente según las especificaciones y comparar con el número de tipos de accesos que se deben registran según las especificaciones	$X = A/B$ A= Número de tipos de accesos que están siendo registrados según las especificaciones B= Número de tipos de accesos que se deben registran según las especificaciones	A se obtiene del código fuente, del diseño y/o del reporte de pruebas de seguridad. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor capacidad del software para auditar los accesos a éste		Revisor de Código y/o Revisor de Pruebas no Funcionales, Revisor de Requerimientos	
Capacidad para controlar los accesos	¿Qué tan controlables son los accesos al sistema?	Contar el número de requerimientos para el control de accesos implementados correctamente según las especificaciones, y comparar con el número de requerimientos para el control de accesos indicados en las especificaciones	$X = A/B$ A= Número de requerimientos para el control de accesos implementados correctamente según las especificaciones. B= Número de requerimientos para el control de accesos indicados en las especificaciones	A se obtiene del código fuente, del diseño y/o del reporte de pruebas de seguridad. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor capacidad del software para controlar los accesos a éste		Revisor de Código y/o Revisor de Pruebas no Funcionales, Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Prevención de corrupción de datos	¿Qué tan completa es la implementación de prevención de corrupción de datos?	Contar el número de instancias de prevención de datos implementadas según lo especificado, y comparar con el número de instancias de operaciones/accesos especificadas en los requerimientos que puedan ser capaz de corrupción y destrucción de datos	$X = A/B$ A= Número de instancias de prevención de datos implementadas según lo especificado. B= Número de instancias de operaciones/accesos identificadas en los requerimientos que puedan ser capaz de corrupción y destrucción de datos	A se obtiene del código fuente, del diseño y/o del reporte de pruebas de seguridad. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor prevención de corrupción de datos		Revisor de Código y/o Revisor de Pruebas no Funcionales, Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Cifrado de datos	¿Qué tan completa es la implementación de cifrados de datos?	Contar el número de instancias para cifrar/descifrar de datos implementadas de acuerdo a las especificaciones, y comparar con el número de instancias de elementos de datos que requieren facilidades para cifrar/descifrar datos según las especificaciones	$X = A/B$ A= Número de instancias para cifrar/descifrar datos implementadas de acuerdo a las especificaciones B= Número de elementos de datos que requieren facilidades para cifrar/descifrar datos según las especificaciones	A se obtiene del código fuente, del diseño y/o del reporte de pruebas de seguridad. B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican que el software permite un mayor cifrado/descifrado de datos según las especificaciones		Revisor de Código y/o Revisor de Pruebas no Funcionales, Revisor de Requerimientos	
Métricas de conformidad de la funcionalidad								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Cumplimiento de estándares entre distintos software	¿Qué tan conforme son las interfaces entre distintos software con la aplicación de regulaciones, normas y convenciones?	Contar el número de interfaces que cumplen con las conformidades requeridas y comparar con el número de interfaces requeridas según las especificaciones de regulaciones, normas y convenciones	$X = A/B$ A= Número de interfaces implementadas correctamente según las especificaciones. B= Número total de interfaces que requieren conformidad según las especificaciones de regulaciones, normas y convenciones	A se obtiene de los prototipos del software, de los prototipos de interfaz no funcionales, y/o de las pruebas funcionales. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor cumplimiento de especificaciones, normas y convenios relacionadas a las interfaces entre distintos software		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Cuadro 1.4: Métricas de Funcionalidad

1.2.3. Métricas de Fiabilidad

Estas métricas son utilizadas para determinar la madurez y rendimiento de un software. Las métricas de fiabilidad se subdividen en:

Métricas de madurez: atributos para evaluar la madurez del software.

Métricas de tolerancia a fallas: atributos para evaluar la capacidad del software para mantener el nivel de rendimiento deseado en caso de fallas operacionales o de incumplimiento de su interfaz especificada.

Métricas de recuperabilidad: atributos para evaluar la capacidad del software de restablecer un nivel adecuado de rendimiento y recuperación de los datos directamente afectados, en caso que ocurra una falla.

Cada una de las sub-métricas indicadas en los ítems anteriores se subdividen a su vez en otras métricas específicas, en función de las cuales se miden atributos de fiabilidad de un software. Estas métricas y la forma en que se miden se describen en el Cuadro 1.5.

Métricas de madurez								
Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Eliminación de fallas	¿Cuál es la proporción de fallas eliminadas?	Contar el número de fallas eliminadas durante el diseño/codificación y comparar con el número de fallas detectadas en las pruebas funcionales	$X = A/B$ A= Número de fallas corregidas en el diseño / codificación. B=Número de fallas detectadas en las pruebas funcionales	El valor de A y B se obtienen del reporte generado en las pruebas funcionales.	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor cantidad de fallas corregidas.		Revisor de Pruebas Funcionales	
Métricas de tolerancia a fallas								
Prevención de fallas	¿Cuántos patrones de fallas fueron puestos bajo control para evitar fallas críticas y graves?	Contar el número de fallas evitadas y comparar con el número de patrones de fallas a ser considerado.	$X = A/B$ A= Número de patrones de fallas que se han evitado en el diseño/código. B= Número de patrones de fallas a ser considerados.	El valor de A se obtiene del reporte de pruebas funcionales. El valor de B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor prevención de fallas		Revisor de Pruebas Funcionales y Revisor de Requerimientos	Ejemplos de patrones de fallas son los puntos muertos (deadlock) fuera del rango de los datos. La técnica de análisis del árbol de fallas puede ser usado para detectar patrones de fallas

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Evitar un funcionamiento incorrecto	¿Cuántas funciones se han implementado con la capacidad de evitar operaciones incorrectas?	Contar el número de funciones implementadas para evitar fallas críticas y graves causadas por operaciones incorrectas realizadas por los usuarios ,y comparar con el número de patrones de operaciones incorrectas a ser consideradas según las especificaciones	$X = A/B$ A= Número de funciones implementadas para evitar patrones incorrectos de operación B= Número de patrones de funcionamiento incorrecto a ser considerados según las especificaciones.	El valor de A se obtiene del código fuente y/o del diseño. El valor de B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Mientras el valor de X se acerque a 1 se evitara mayor cantidad de operaciones incorrectas		Revisor de Código y/o Revisor de Arquitectura del Software, Revisor de Requerimientos	
Métricas de recuperabilidad								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Restaurabilidad	¿Qué tan capaz es la aplicación de software de restaurarse por si misma después de un evento anormal o de una petición?	Contar el número de requisitos de restauración implementados y comparar con el número de requisitos de restauración especificados.	$X = A/B$ A= Número de requerimientos de restauración implementados. B= Número de requerimientos de restauración especificados	El valor de A se obtiene del código fuente y/o del reporte de pruebas funcionales. El valor de B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor grado de restauración del sistema ante eventos anormales o peticiones determinadas		Revisor de Código y/o Revisor de Pruebas Funcionales, Revisor de Requerimientos	Ejemplos de requerimientos de restauración: Puntos de verificación (checkpoint) de bases de datos, puntos de verificación (checkpoint) de transacciones, la función deshacer y la función rehacer. Existencia de un respaldo de la base de datos, con lo cual el software se actualice automáticamente cada cierto tiempo de acuerdo a lo especificado, en caso de ocurrir una falla, el software puede recuperar los datos.

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Efectividad de la restauración	¿Qué tan efectiva es la capacidad de restauración?	Contar el número de requerimientos de restauración con tiempos objetivos implementados (con cálculos o simulaciones), y comparar con el número de requerimientos de restauración con tiempos objetivos especificados.	$X = A/B$ A= Número de requerimientos de restauración implementados que hayan cumplido con el tiempo objetivo especificado. B= El número de requerimientos de restauración con tiempos objetivos especificados.	A se obtiene de los reportes de pruebas funcionales. B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor efectividad de restauración del software.		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Cuadro 1.5: Métricas de Fiabilidad

1.2.4. Métricas de Usabilidad

Estas métricas son utilizadas para medir el esfuerzo que deben realizar los usuarios para aprender a usar el software. La usabilidad puede ser medida en función de varias sub-métricas, para el caso de este modelo se consideran las siguientes:

Capacidad para ser entendido: son utilizadas para evaluar la capacidad del software para permitir a los usuarios entender si éste es adecuado, y cómo este puede ser utilizado para determinadas tareas y bajo ciertas condiciones de uso.

Capacidad para ser aprendido: son utilizadas para evaluar la capacidad del software para permitir a los usuarios aprender a usar las funciones que éste brinda.

Capacidad para ser operado: son utilizadas para evaluar la capacidad del software para permitir a los usuarios que lo operen y lo controlen.

Capacidad de atracción: son utilizadas para evaluar la apariencia del software en términos de que éste resulte atractivo a los usuarios.

Cumplimiento de estándares: son utilizadas para evaluar la capacidad del software para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

Efectividad: son utilizadas para evaluar la exactitud y la exhaustividad con la que los usuarios pueden alcanzar sus objetivos.

Eficiencia: son utilizadas para evaluar el esfuerzo que deben realizar los usuarios para alcanzar sus metas.

Satisfacción: son utilizadas para evaluar cómo se sienten los usuarios con el software.

Calidad de los canales de atención al usuario (soporte técnico): son utilizadas para evaluar la calidad de los canales de atención al usuario, respecto a consultas que éstos realicen relacionadas con el funcionamiento e instalación del software.

Documentación de usuarios: son utilizadas para evaluar la calidad de los manuales de usuario (archivos de ayuda en linea, screencasts, demos, tutoriales, entre otros.).

Cada una de las sub-métricas indicadas en los ítems anteriores a su vez se subdividen en otras métricas específicas, en función de las cuales se miden atributos de usabilidad de un software. Estas métricas y la forma en que éstas se miden se describen en el Cuadro 1.6.

Métricas de capacidad para ser entendido								
Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Descripción completa	¿Qué proporción de funciones (o tipos de funciones) se encuentran en la descripción del software?	Contar el número de funciones adecuadamente descritas y comparar con el total de funciones del software	$X = A/B$ A= Número de funciones (o tipo de funciones) que se encuentran en la descripción del software. B= Número total de funciones (o tipo de funciones).	A se obtiene del manual de usuario. B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores cercanos a 1 indican una descripción más completa de las funciones del software.		Revisor de Manuales del software y Revisor de Requerimientos	
Capacidad de demostración	¿Qué proporción de funcionalidades que requieren demostración tienen demostración?	Contar el número de funciones que están adecuadamente demostradas y comparar con el total de funciones que requieren demostración.	$X = A/B$ A= Número de funciones demostradas en la descripción del software. B= Número total de funciones que requieren demostración.	A se obtiene del manual de usuario. B se obtiene de la especificación de requerimientos y/o del diseño.	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de funciones del software que requieren demostración y son demostradas.		Revisor de Manuales del software y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Funciones evidentes	¿Qué proporción de las funciones del software son evidentes al usuario?	Contar las funciones evidentes al usuario y comparar con el número total de funciones.	$X = A/B$ A= Número de funciones (o tipos de funciones) evidentes al usuario. B= Total de funciones (o tipos de funciones).	A se obtiene de los prototipos del software. B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de funciones del software que son evidentes al usuario.		Revisor de Manuales del software y Revisor de Requerimientos.	Esta métrica indica si los usuarios serán capaces de encontrar las funciones explorando al interfaz (ejemplo: inspeccionando el menú).
Función de entendibilidad	¿Qué proporción de las funciones del software sera capaz de entender el usuario?	Contar el número de funciones cuyos propósitos son entendidos por los usuarios y comparar con el número de funciones de la interfaz.	$X = A/B$ A= Número de funciones de la interfaz cuyos propósitos son entendidos por el usuario. B= Número de funciones de la interfaz.	A se obtiene de las opiniones de los usuarios y/o de los probadores del software. B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de funciones cuyos propósitos son entendidos por el usuario.		Revisor de Requerimientos	Posibles estrategias para medir esta métrica: talleres con usuarios, rondas entre el equipo de aseguramiento de calidad, entre otras.
Métricas de capacidad para ser aprendido								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Existencia completa de la documentación de usuario y/o ayudas	¿Qué proporción de funciones son descritas en la documentación y/o ayudas?	Contar el número de funciones implementadas con ayudas y/o documentación y comparar con el número total de funciones en el software	$X = A/B$ A= Número de funciones que se encuentran descritas. B= Número total de funciones.	A se obtiene del manual de usuarios. B se obtiene de la especificación de requerimientos.	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de funciones del software descritas en la documentación		Revisor de Manuales del software y Revisor de Requerimientos	
Métricas de capacidad para ser operado								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Validación de datos de entrada	¿Qué proporción de campos chequean la validez de los datos?	Contar el número de campos que chequean la validez de los datos de entrada, y comparar con el número de campos total que deberían chequear la validez los datos	$X = A/B$ A= Número de campos que validan los datos de entrada. B= Número total de campos que deberían validar los datos de entrada	A se obtiene de las pruebas funcionales. B se obtiene de la especificación de requerimientos (para que la especificación pueda ser utilizada aquí tiene que indicarse en esta especificación el número total de campos que deben validar los datos de entrada)	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de datos de entrada validados		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad del usuario de cancelar operaciones en el software	¿Qué proporción de funciones pueden ser canceladas antes de ser completadas?	Contar el número de funciones implementadas que pueden ser canceladas por el usuario antes de completarlas, y comparar con el número de funciones que requieren la capacidad de ser canceladas antes de ser completadas	$X = A/B$ A= Número de funciones implementadas que pueden ser canceladas por el usuario. B= Número de funciones que requieren la capacidad de ser canceladas antes de ser completadas	A se obtiene de las pruebas funcionales y/o de los prototipos del software. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de funciones del software que requieren capacidad de cancelación y que pueden ser efectivamente canceladas antes de ser completadas		Revisor de Pruebas Funcionales y Revisor de Requerimientos	
Capacidad del usuario para deshacer	¿Qué proporción de funciones pueden ser deshechas?	Contar el número de funciones implementadas que pueden deshechas por el usuario después de ser completadas, y comparar con el número total de funciones del software	$X = A/B$ A= Número de funciones implementadas que pueden ser deshechas por el usuario. B= Número total de funciones del software	A se obtiene de las pruebas funcionales y/o de los prototipos del software. B se obtiene de la especificación de requerimientos y/o del diseño	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad del funciones del software que pueden ser deshechas por el usuario después de ser completadas		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Personalización	¿Qué proporción de funciones pueden ser personalizadas?	Contar el número de funciones implementadas que pueden ser personalizadas por el usuario durante la operación, y comparar con el número de funciones que requieren la capacidad de personalización	$X = A/B$ A= Número de funciones que pueden ser personalizadas durante las operaciones. B= Número de funciones que requieren la capacidad de personalización	A se obtiene de las pruebas funcionales y/o de los prototipos del software. B se obtiene de la especificación de requerimientos y/o del diseño	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor capacidad de personalización del software		Revisor de Pruebas Funcionales y Revisor de Requerimientos	
Accesibilidad física	¿Qué proporción de funciones podrían ser personalizadas para ser accedidas por usuarios con discapacidades físicas?	Contar el número de funciones implementadas que podrían llegar a ser personalizadas y comparar con el número total de funciones	$X = A/B$ A= Número de funciones que podrían ser personalizadas. B= Número total de funciones del software	A se obtiene de las pruebas funcionales y/o de los prototipos del software. B se obtiene de la especificación de requerimientos y/o del diseño	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor capacidad de accesibilidad física del software		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad de supervisión del estado de las operaciones	¿Qué proporción de funciones son capaces de supervisar el estado de sus operaciones?	Contar el número de funciones implementadas cuyo estado de operaciones puede ser supervisado, y comparar con el número de funciones que requieren esta capacidad de supervisión	$X = A/B$ A= Número de funciones que poseen la capacidad de supervisar el estado de sus operaciones. B= Número de funciones que requieren la capacidad de supervisión	A se obtiene de los prototipos del software y/o del reporte de pruebas funcionales. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor capacidad de supervisión de las funciones del software que requieren esta capacidad		Revisor de Pruebas Funcionales y Revisor de Requerimientos	
Consistencia operacional	¿Qué proporción de operaciones se comportan de la misma manera que operaciones similares en otras partes del software?	Contar número de operaciones o funciones que deberían tener comportamientos similares y comparar con el número de operaciones o funciones que deben tener comportamientos similares	$X = 1 - A/B$ A= Número de operaciones que deberían tener comportamientos similares pero tiene comportamientos distintos. B= Número total de operaciones que deben tener comportamientos similares	A se obtiene de las pruebas funcionales y/o de los prototipos del software. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor consistencia operacional del software		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Claridad de los mensajes	¿Qué proporción de los mensajes se explican por si mismos?	Contar el número de mensajes implementados con una explicación clara, y comparar con el número total de mensajes implementados	$X = A/B$ A= Número de mensajes implementados con explicaciones claras. B= Número de mensajes implementados	A y B se obtienen de los prototipos del software	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor claridad en los mensajes presentados por el software		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Claridad de los funciones de la interfaz	¿Qué proporción de funciones de la interfaz se explican por si mismas con descripciones textuales?	Contar el número de funciones de la interfaz que se se explican por si mismas y compararlo con el número total de funciones de la interfaz	$X = A/B$ A= Número de funciones de la interfaz que se se explican por si mismos. B= Número de funciones de la interfaz	A y B se obtienen de los prototipos del software	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de funciones de la interfaz que se explican por si mismas		Revisor de Pruebas Funcionales y Revisor de Requerimientos	Las funciones se explican por si mismas cuando ellas usan texto plano o cuando cuentan con hover-help o tool tips. Tool tip: es una corta descripción de un campo en forma textual, reporte o título. También es llamado hover help. Para observar esta clase de ayuda se coloca el cursor sobre el elemento de la interfaz y una breve descripción del elemento aparecerá en un recuadro al lado del cursor

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Métricas de capacidad de atracción								
Interacción atractiva	¿Qué tan atractiva es la interfaz para el usuario?	Cuestionario al usuario. Este cuestionario se realizará al usuario para evaluar el atractivo de la interfaz, tomando en cuenta atributos como: colores, diseños gráficos. Cada pregunta será evaluada de acuerdo a la siguiente escala de intervalos entre el cero (0) y uno (1): a) (0,8 - 1] ->Muy Alto b) (0,6 – 0,8] ->Alto c) (0,4 – 0,6] ->Medio d) (0,2 – 0,4] ->Bajo e) [0 – 0,2] ->Muy Bajo	$X = \Sigma A_i / B$ $A_i = \Sigma V_j / P$ V_j= Valor asignado por el $Usuario_i$ a la $pregunta_j$ P= Número de preguntas realizadas al $Usuario_i$ A_i= Valor ponderado de las respuestas del $Usuario_i$ B= Número de usuarios que respondieron el cuestionario	Opinión de los usuarios	$0 \leq X \leq 1$ Valores cercanos a 1 indican una interfaz más atractiva para el usuario		Revisor de Pruebas Funcionales	Aspectos que potencialmente pueden contribuir al atractivo incluyen: alineación de los items (vertical u horizontal), agrupaciones, uso de los colores, uso apropiado y racional de los tamaños de los gráficos, uso de espaciados, separadores y/o bordes, tipografías, e interfaces 3D

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad de personalización de la interfaz del usuario	¿Qué proporción de elementos de la interfaz del usuario pueden ser personalizados en apariencias?	Inspección (por un experto)	$X = A/B$ A= Número de tipos o elementos de la interfaz que se pueden personalizar B= Número de tipos o elementos de la interfaz	A se obtiene de los prototipos del software. B se obtiene de los prototipos no funcionales de la interfaz	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor capacidad de personalización de los elementos de la interfaz de usuario		Revisor de Pruebas Funcionales y Revisor de Requerimientos	Es importante tener presente que pueden existir aplicaciones en las cuales no se permita la personalización de algunos o todos los elementos de la interfaz, motivado por ejemplo a razones de seguridad, entre otras
Métricas de cumplimiento de estándares								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Cumplimiento de estándares	¿Qué tan ajustado está el software a las regulaciones, estándares y convenciones de usabilidad aplicables?	Contar el número de estándares que se han cumplido y comparar con el número de estándares aplicables al software	$X = A/B$ A= Número de estándares de usabilidad que se cumplen. B= Número de estándares de usabilidad aplicables al software	A se obtiene de los prototipos del software B se obtiene de los estándares	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cumplimiento de estándares de usabilidad		Revisor de Pruebas Funcionales y Revisor de Requerimientos	Para medir esta métrica se recomienda realizar una lista de los estándares aplicables y marcar cuando éstos se cumplen
Métricas de Efectividad								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Tareas completadas	¿Qué proporción de tareas llegan a ser realizadas por el usuario?	En la prueba de usabilidad medir el número de funciones completadas	$X = \Sigma A_i/B$ A_i= Número de tareas completadas por el $Usuario_i$ B= Número de tareas establecidas, multiplicado por el número total de usuarios	A y B se obtienen del reporte de resultados de la prueba. Este reporte debe contener: Nº total de usuarios, Nº de tareas establecidas y Nº de tareas completadas por usuario	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor cantidad de tareas completadas por los usuarios definidos para la prueba		Revisor de Pruebas Funcionales	Para estas métricas se debe llevar a cabo una prueba de usabilidad en donde se establecen objetivos o tareas que se consideren de gran importancia (incluir tareas relacionadas con los medios de atención al usuario y con el uso de la documentación del software), y se debe seleccionar un número total de usuarios que se considere pertinente para llevar a cabo dichas tareas. Los expertos recomiendan de 20 a 40 usuarios, pero dicho número

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Usuarios capaces de completar las tareas	¿Qué proporción de usuarios son capaces de completar con éxito las tareas establecidas?	En la prueba de usabilidad medir el número de usuarios que cumplieron el total de funciones establecidas	$X = A/B$ A= Número de usuarios que completaron todas las tareas establecidas? B= Número total de usuarios.	A y B se obtienen del reporte de resultados de la prueba. Este reporte debe contener: Nº total de usuarios y Nº de usuarios que completaron todas las tareas establecidas	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor facilidad por parte de los usuarios para utilizar el software		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Errores cometidos durante la realización de las tareas establecidas	¿Qué proporción de errores son cometidos durante la prueba?	En la prueba de usabilidad medir el número de errores cometidos	$X = 1 - A/B$ A= Número de errores cometidos durante la prueba. B= Número de tareas establecidas, multiplicado por el número total de usuarios	A y B se obtienen del reporte de resultados de la prueba. Este reporte debe contener: N° total de usuarios, N° de tareas establecidas y N° de errores cometidos durante la prueba	$-\infty \leq X \leq 1$ Valores cercanos a 1 indican menos errores cometidos en la prueba realizada		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Número de solicitudes de ayuda para llevar a cabo las tareas establecidas	¿Qué proporción de solicitudes de ayuda son realizadas durante la prueba?	En la prueba de usabilidad medir el número de veces que los usuarios solicitan ayuda para realizar las tareas	$X = 1 - A/B$ A= Número de solicitudes de ayuda B= Número de tareas establecidas, multiplicado por el número de usuarios total	A y B se obtienen del reporte de resultados de la prueba. Este reporte debe contener: Nº total de solicitudes de ayuda, Nº de tareas establecidas y Nº total de usuarios	$\infty \leq X \leq 1$ Valores cercanos a 1 indican que el software puede ser utilizado con mayor facilidad		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Porcentaje de usuarios que pueden llevar a cabo las tareas sin leer el manual	¿Qué proporción de usuarios llevan a cabo las tareas establecidas sin consultar el manual?	En la prueba de usabilidad medir el número de usuarios que realizan las tareas sin consultar el manual	$X = A/B$ A= Número de usuarios que no consultaron el manual durante la prueba. B= Número total de usuarios	A y B se obtienen del reporte de resultados de la prueba. Este reporte debe contener: Nº de usuarios que consultaron el manual y Nº total de usuarios	$0 \leq X \leq 1$ Valores cercanos a 1 indican que mayor cantidad de usuarios pueden utilizar el software sin necesidad de consultar el manual de usuarios		Revisor de Pruebas Funcionales	
Métricas de Eficiencia								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Tiempo para realizar tareas en comparación con un experto	¿Qué diferencia de tiempo necesitan los usuarios para realizar las tareas en relación con un experto?	En la prueba de usabilidad medir el tiempo que cada usuario tarda en realizar todas las tareas y promediarlo, luego comparar con el tiempo que le toma a un experto realizar la misma prueba	$X = A - B$ A= Tiempo que le toma a un experto completar la prueba. B= Tiempo promedio que le toma a los usuarios completar la prueba: sumatoria de los tiempos de cada usuario dividido entre el número total de usuarios	A y B se obtienen del reporte de resultados de la prueba. Este reporte debe contener: tiempo que usó el experto para completar la prueba, tiempo promedio que le toma a los usuarios completar la prueba y Nº total de usuarios	$-\infty \leq X \leq +\infty$ Valores positivos de X indican que el software es más fácil de utilizar. El rango de valores negativos aceptables será definido por el experto		Revisor de Pruebas Funcionales	Estas métricas se miden en función de la prueba de usabilidad indicada para las métricas de efectividad

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Tiempo para instalar el software	¿Qué diferencia de tiempo necesitan los administradores para instalar el software en relación con un experto?	Medir el tiempo promedio que le toma a distintos administradores instalar el software y comparar con el tiempo que le toma a un experto instalarlo	$X = A - B$ A= Tiempo que le toma a un experto instalar el software. B= Tiempo promedio que le toma a los administradores del software instalar el software: sumatoria de los tiempos de cada administrador entre el número total de administradores	Instalación de software por los administradores y el experto	$-\infty \leq X \leq +\infty$ Valores positivos de X indican que el software es más fácil de instalar. El rango de valores negativos aceptables será definido por el experto		Revisor de Código	
Métricas de Satisfacción								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Actitud positiva ante el software	¿Qué proporción de los usuarios tienen una actitud positiva con respecto al software?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿cómo evaluaría su actitud positiva frente al software? La pregunta será evaluada de acuerdo a la siguiente escala de intervalos entre el cero (0) y uno (1): a) (0,8 - 1] ->Muy Alto b) (0,6 – 0,8] ->Alto c) (0,4 – 0,6] ->Medio d) (0,2 – 0,4] ->Bajo e) [0 – 0,2] ->Muy Bajo	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican más actitudes positivas de parte de los usuarios ante el software		Revisor de Pruebas Funcionales	Las preguntas planteadas para cada una de las métricas de satisfacción se deben realizar a los usuarios que participaron en la prueba de usabilidad indicada para las métricas de eficiencia y efectividad, mostradas anteriormente. Estas preguntas se realizan una vez culminada la prueba respectiva

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Satisfacción de los usuarios del software en comparación con el software anterior	¿Qué proporción de los usuarios encuentran el software más satisfactorio que el anterior que utilizaban para las mismas tareas?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿cuál es su grado de satisfacción con el software al compararlo con el software anterior que utilizaba para las mismas tareas? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para la métrica anterior	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor satisfacción de los usuarios con respecto al software		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Percepción de control con respecto al software	¿Qué proporción de usuarios se sienten .[en] control"del software?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿cuál es su percepción de control sobre el software? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para la métrica anterior.	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor control de los usuarios sobre el software		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Aporte a la productividad por el uso del software	¿Qué tanto ayuda el software a la productividad de los usuarios?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿aumenta su productividad respecto a las tareas que realiza con el software? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para la métrica anterior	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor productividad de los usuarios al utilizar el software		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Percepción de que el software apoya las tareas según sea necesario por el usuario	¿Qué tanto apoya el software a la tareas de los usuarios?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿que tanto apoyo presta el software a las tareas que usted realiza? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para la métrica anterior	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican que el software apoya en mayor grado las tareas que realizan los usuarios		Revisor de Pruebas Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Recomendación del software a otros usuarios	¿Qué tan recomendable es el software?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿considera el software recomendable a otros usuarios? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para la métrica anterior	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican que el software es recomendable a otros usuarios		Usuarios Revisor de Pruebas Funcionales	
Métricas de Calidad de los canales de atención al usuario (soporte técnico)								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Calidad de los medios de atención al usuario	¿Qué proporción de usuarios se encuentran satisfechos con los servicios prestados por los medios de atención al usuario?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿se encuentra satisfecho con los servicios prestados por los medios de atención al usuario? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para las métricas de satisfacción	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican buena calidad de los medios de atención al usuarios		Revisor de Manuales del Software	Las preguntas planteadas para las métricas de canales de atención al usuario deben realizarse a los usuarios que participaron en la prueba de usabilidad indicada para las métricas de eficiencia y efectividad, una vez culminada la prueba.[8]
Métricas de Documentación de usuarios								

[8]Los canales de atención al usuario pueden ser: números de teléfono, correos electrónicos para soporte a usuarios, lista de discusión de usuarios, canales IRC (Internet Relay Chat) para soporte en línea, listado de preguntas frecuentes como FAQ (Frequently Asked Questions), sistemas de tickets para el reporte y seguimiento de errores reportados por los usuarios, entre otros.

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Objetivos de la documentación	¿Qué proporción de usuarios indica que la documentación de usuarios refleja los objetivos que se persiguen con la misma?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿la estructura de la documentación de usuarios refleja los objetivos que persigue dicha documentación? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para las métricas de satisfacción	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican que la estructura de la documentación de usuarios refleja con mayor grado los objetivos que se persiguen con la misma		Revisor de Manuales del Software	Las preguntas planteadas para las métricas de documentación de usuarios deben realizarse a los usuarios que participaron en la prueba de usabilidad indicada para las métricas de eficiencia y efectividad, una vez culminada la prueba

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Claridad de la documentación	¿Qué proporción de usuarios indica que la documentación de usuarios es sencilla y clara?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿la documentación de los usuarios se encuentra descrita de manera sencilla y clara? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para las métricas de satisfacción	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican que la documentación de usuarios está descrita de forma más clara y sencilla		Revisor de Manuales del Software	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Acceso directo	¿Es posible acceder de manera selectiva a partes de la información indicada en la documentación de usuarios?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿considera que es posible acceder a partes de la información indicada en la documentación de usuarios de manera selectiva? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para las métricas de satisfacción	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor grado de acceso directo a partes de la información indicada en la documentación de usuarios		Revisor de Manuales del Software	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Terminología utilizada	¿Los términos utilizados en la documentación de usuarios son similares a los términos utilizados en la interfaz de usuarios?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿considera que los términos utilizados en la documentación de usuarios son similares a los términos utilizados en la interfaz de usuarios del software? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para las métricas de satisfacción	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor similitud entre los términos utilizados en la documentación de usuarios y los términos utilizados en la interfaz de usuarios		Revisor de Manuales del Software	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Consistencia	¿Existe coherencia entre los puntos tratados en cada una de las secciones de la documentación?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿considera que existe coherencia entre los puntos tratados en cada una de las secciones de la documentación de usuarios? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para las métricas de satisfacción	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada. B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican mayor consistencia entre los puntos tratados en cada una de las secciones de la documentación de usuarios		Revisor de Manuales del Software	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Diseño	¿Apoya el diseño de la documentación de usuarios la exploración y los saltos entre las secciones de esta documentación?	Realizar la siguiente pregunta a cada uno de los usuarios participantes en la prueba: ¿considera que el diseño de la documentación de usuarios permite la exploración y los saltos entre las distintas secciones que conforman esta documentación? Para contestar esta pregunta el usuario debe utilizar la misma escala de valor indicada para las métricas de satisfacción	$X = A/B$ A= Sumatoria de los valores cuantitativos dados por los usuarios para la pregunta realizada B= Número total de usuarios	Opinión de los usuarios respecto a la pregunta realizada	$0 \leq X \leq 1$ Valores cercanos a 1 indican que el diseño de la documentación permite en mayor grado la exploración y los saltos entre las secciones de la documentación		Revisor de Manuales del Software	

Cuadro 1.6: Métricas de Usabilidad

1.2.5. Métricas de Eficiencia

Estas métricas son usadas para evaluar el rendimiento del software en función de la cantidad de recursos utilizados por éste. Estas métricas se miden en base a ciertas características presentes en el software, como tiempos de respuesta y de procesamiento, cantidad de recursos utilizados, entre otras. En base a estas características se han definido un conjunto de sub-métricas relacionadas a la eficiencia del software, entre las cuales las más utilizadas son:

Métricas de comportamiento en el tiempo: son utilizadas para evaluar la capacidad del software para proporcionar tiempos de respuesta y de procesamiento apropiados bajo condiciones determinadas.

Métricas de utilización de recursos: son utilizadas para medir la cantidad y tipo de recursos que utiliza un software operando bajo determinadas condiciones.

Cada una de estas sub-métricas se dividen a su vez en un conjunto de métricas específicas que permiten medir con mayor facilidad las características referidas a la eficiencia de un software. Estas métricas y la forma en que éstas se miden se describen en la Tabla 1.7.

Métricas de comportamiento en el tiempo								
Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Tiempo de respuesta para una tarea específica	¿Cuál es el tiempo estimado para completar una tarea específica?	Evaluar la eficacia de las llamadas al sistema operativo y a la aplicación de software. Estimar el tiempo de respuesta de la aplicación de software basado en lo anterior. Para este caso se puede medir: todas o parte de las especificaciones de diseño, probar la ruta completa de una transacción, probar módulos o partes completas del software, producto de software completo durante la fase de pruebas no funcionales.	$X = Tiempo$ (calculado o simulado)	Sistema operativo. Tiempo estimado de respuesta del sistema ante una solicitud (este tiempo se obtiene en las pruebas no funcionales)	Valores pequeños de X indican tiempos de respuesta más cortos		Revisor de Pruebas no Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Tareas por unidad de tiempo	¿Cuál es el número estimado de tareas que pueden ser realizadas por encima de una unidad de tiempo?	Evaluar la eficiencia de manejo de los recursos en el sistema. Establecer un factor basado en las llamadas a la aplicación de software para el manejo de recursos	X = Número de tareas por encima de una unidad de tiempo	Sistema operativo. Tiempo estimado de tareas realizadas por el software, por encima de una unidad de tiempo (este tiempo se obtiene en las pruebas no funcionales)	Valores mayores de X indican mayor cantidad de tareas realizadas por el software por encima de una unidad de tiempo		Revisor de Pruebas no Funcionales	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Tiempo de respuesta para tareas que involucran trabajo en lote	¿Cuál es el tiempo estimado para completar un grupo de tareas relacionadas que involucran un trabajo en lote?	Evaluar la eficacia del sistema operativo y las llamadas a la aplicación. Estimar el tiempo de respuesta para completar un grupo de tareas relacionadas basado en lo anterior. En este caso se puede medir: todas o parte de las especificaciones de diseño, probar la ruta completa de una transacción, probar módulos o partes completas del software, producto de software completo durante la fase de pruebas no funcionales	$X = Tiempo$ (calculado o simulado)	Sistema operativo. Tiempo estimado en el que el software completa un grupo de tareas relacionadas (este tiempo se obtiene en las pruebas no funcionales)	Valores menores de X indican que el software realiza en tiempos más cortos tareas que involucran trabajo en lote		Revisor de Pruebas no Funcionales	Esta métrica se aplica cuando la aplicación de software tiene tareas por lote
Métricas de utilización de recursos								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Utilización de memoria	¿Cual es el tamaño de memoria estimado que ocuparía la aplicación de software para completar una tarea específica?	Estimar el requerimiento de memoria	X = Tamaño en bytes (calculado o simulado)	Estimar el tamaño de utilización de memoria (éste se obtiene en las pruebas no funcionales)	Valores menores de X indican que el software utiliza menor cantidad de memoria para completar una tarea específica		Revisor de Pruebas no Funcionales	

Cuadro 1.7: Métricas de Eficiencia

1.2.6. Métricas de Mantenibilidad

Estas métricas son utilizadas para medir el nivel de esfuerzo requerido para modificar el software. La mantenibilidad puede ser medida en función de varias sub-métricas, para el caso de este modelo se consideran las siguientes:

Métricas de capacidad de análisis: indican un conjunto de atributos para predecir el esfuerzo y recursos requeridos por los usuarios o mantenedores para diagnosticar las deficiencias o causas de fallas, o para identificar las partes que deben ser modificadas en el software.

Métricas de capacidad para ser modificado: indican el conjunto de atributos para predecir el esfuerzo requerido por el personal de mantenimiento o usuarios para implementar una modificación especificada en el software. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software a los cambios en el entorno y en los requerimientos. La implementación incluye los cambios en el diseño, el código y la documentación.

Métricas de estabilidad: indican el conjunto de atributos para medir la capacidad del software para evitar efectos inesperados debido a modificaciones del software.

Cada una de estas sub-métricas se dividen a su vez en un conjunto de métricas específicas que permiten medir con mayor facilidad las características referidas a la mantenibilidad de un software. Estas métricas y la forma en que éstas se miden se describen en el Cuadro 1.8

Métricas de capacidad de análisis								
Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Registro de actividades	¿Cuán completo o exhaustivo es el registro del estado del software?	Contar el número de elementos registrados de una sesión de usuario y comparar con el número de elementos que requieren ser registrados según las especificaciones	$X = A/B$ A= Número de elementos registrados en una sesión de usuario. B= Número total de elementos que deberían ser registrados de acuerdo a las especificaciones	El valor de A se obtiene del conteo de los elementos registrados en el historial del software. El valor de B se obtiene de las especificaciones de los requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican que se registran mayor cantidad de elementos en el historial de registros de la aplicación de software.		Revisor de Código y Revisor de Requerimientos	El registro donde se almacena la información de la sesión de un usuario es el "Historial de registros de la aplicación de software". Esta métrica tiene que ser medida para cada tipo de usuario de la aplicación de software.

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Disponibilidad de las funciones de diagnóstico	¿Cuán completa es la provisión de las funciones de diagnóstico?	Contar el número de funciones de diagnóstico implementadas de acuerdo con las especificaciones y comparar con el número de funciones de diagnóstico requeridas según las especificaciones	$X = A/B$ A= Número de funciones de diagnóstico implementadas de acuerdo a las especificaciones. B= Número de funciones de diagnóstico requeridas según las especificaciones	El valor de A se obtiene del código fuente y/o de las pruebas funcionales. El valor de B se obtiene de las especificaciones de los requerimientos.	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor disponibilidad de las funciones de diagnóstico implementadas de acuerdo con las especificaciones		Revisor de Código y/o Revisor de Pruebas Funcionales, Revisor de Requerimientos.	Las funciones de diagnóstico sirven para que el usuario pueda verificar el funcionamiento de los módulos de un software, por ejemplo, si están activos o inactivos
Métricas de capacidad para ser modificado								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Registro de cambios	¿Los cambios realizados a las especificaciones y módulos del software son registrados adecuadamente en el código y haciendo uso de comentarios?	Contar el número de cambios en funciones o módulos del software que tienen comentarios, y comparar con el número de funciones o módulos alterados desde la primera versión.	$X = A/B$ A= Número de cambios en funciones y/o módulos que tienen comentarios. B= Número total de funciones y/o módulos alterados desde la primera versión del software	El valor de A se obtiene del código fuente. El valor de B se obtiene del registro de control de cambios y de versiones y del código fuente (en caso de que en el control de cambios no se especifiquen todos los cambios realizados).	$0 \leq X \leq 1$ Valores de X más cercanos a 1 indican mayor registro de cambios en el software.		Revisor de Código	
Métricas de estabilidad								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Impacto de cambios	¿Que tan fuertes son los impactos generados en una versión del software por modificaciones que se realicen sobre ésta?	Contar el número de impactos adversos detectados después de las modificaciones y comparar con el número de modificaciones realizadas	$X = A/B$ A= Número de impactos adversos detectados después de las modificaciones. B= Número de modificaciones realizadas	El valor de A se obtiene del reporte de pruebas funcionales. B se obtiene del registro de control de cambios y versiones y/o del código fuente	$0 \leq X$ Valores de X cercanos a 0 indican menos impactos que se generan en el software debido a modificaciones realizadas sobre éste		Revisor de Pruebas Funcionales y Revisor de Código	

Cuadro 1.8: Métricas de Mantenibilidad

1.2.7. Métricas de Portabilidad

Estas métricas son utilizadas para medir el esfuerzo necesario para transferir el software de un entorno de sistema hardware y/o software a otro entorno diferente. La portabilidad puede ser medida en función de varias sub-métricas, para el caso de este modelo se consideran las siguientes:

Métricas de adaptabilidad: indican el conjunto de atributos para medir la capacidad del software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software considerado.

Métricas de facilidad para ser instalado: indican el conjunto de atributos para medir la capacidad del software para ser instalado en un entorno especificado.

Métricas de coexistencia: indican el conjunto de atributos para medir la capacidad del software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.

Cada una de estas sub-métricas se dividen a su vez en un conjunto de métricas específicas que permiten medir con mayor facilidad las características referidas a la portabilidad de un software. Estas métricas y la forma en que éstas se miden se describen en el Cuadro 1.9.

Métricas de adaptabilidad								
Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad para adaptar las estructuras de datos	¿Cuán adaptable es el software a cambios en las estructuras de datos?	Contar el número de estructuras de datos que son operables y no tienen limitaciones después de la adaptación y comparar con el número total de estructuras de datos	$X = A/B$ A= Número de las estructuras de datos que son operables y no tienen limitaciones después de la adaptación. B= Número total de las estructuras de datos	A se obtiene de los reportes de pruebas funcionales y/o del código fuente. B se obtiene del diseño de base de datos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor capacidad de software para adaptar las estructuras de datos		Revisor de Datos Persistentes	
Capacidad para ser adaptado al entorno del hardware (capacidad para ser adaptado a dispositivos de hardware e instalaciones de redes)	¿Cuán adaptable es el software a los cambios del entorno relacionados al hardware?	Contar el número de funciones implementadas que son capaces de alcanzar resultados en entornos de hardware múltiples especificados y comparar con el número total de funciones del software	$X = A/B$ A= Número de las funciones implementadas que son capaces de alcanzar resultados en múltiples entornos de hardware según lo especificado. B= Número total de funciones del software	A se obtiene de los resultados de las pruebas funcionales. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor capacidad del software para adaptarse a entornos de hardware		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad para ser adaptado al entorno organizacional (capacidad para ser adaptado a la organización y a la infraestructura de la misma)	¿Cuán adaptable es el software al cambio organizacional ?	Contar el número de las funciones implementadas que son capaces de alcanzar los resultados requeridos en organizaciones múltiples según lo especificado y comparar con el número total de funciones del software	$X = A/B$ A= Número de las funciones implementadas que son capaces de alcanzar los resultados requeridos en los ambiente de organizaciones y de negocio múltiples según lo especificado. B= Número total de funciones del software	A se obtiene del reporte de pruebas funcionales. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor capacidad del software para adaptarse a entornos organizacionales		Revisor de Pruebas Funcionales y Revisor de Requerimientos	

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Facilidad de portabilidad para el usuario	¿Cuánto esfuerzo es necesario para realizar operaciones portables al software?	Contar el número de las funciones implementadas para facilitar al usuario utilizar el software desde otros entornos (software y hardware), y comparar con el número de funciones especificadas para facilitar al usuario utilizar el software desde otros entornos (software y hardware)	$X = A/B$ A= Número de las funciones implementadas para facilitar al usuario utilizar el software desde otros entornos. B= Número de funciones especificadas para facilitar al usuario utilizar el software desde otros entornos (software y hardware)	A se obtiene del reporte de pruebas funcionales y no funcionales. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor facilidad de portabilidad del software		Revisor de Pruebas Funcionales y Revisor de Requerimientos	Ejemplo de funcionalidad de portabilidad para el usuario: usuario que desea conectarse a la aplicación software desde un navegador web o desde un dispositivo móvil, lo cual evita que el usuario tenga que trasladarse a un lugar específico para hacer uso del software

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad para ser adaptado al entorno del software (al sistema operativo, al software de redes, al software de la aplicación instalada y a navegadores web (en caso de ser una aplicación web))	¿Cuán adaptable es el software a los cambios de entorno relativos al sistema operativo, al software de redes, al software de la aplicación instalada y a navegadores web	Contar el número de funciones implementadas que son capaces de alcanzar los resultados requeridos en entornos múltiples de sistemas de software según lo especificado, y comparar con el número de funciones con requisitos de capacidad de adaptación a múltiples sistemas de software	$X = A/B$ A= Número de funciones implementadas que son capaces de alcanzar los resultados requeridos en entornos múltiples de sistemas de software según lo especificado. B= Número total de funciones con requisitos de capacidad de adaptación a múltiples sistemas de software	A se obtiene del reporte de pruebas funcionales y/o del código fuente. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor capacidad del software para ser adaptado a cambios de entorno relativos al sistema operativo, al software de redes, al software de la aplicación instalada y a navegadores web		Revisor de Pruebas Funcionales y Revisor de Requerimientos	
Métricas de facilidad para ser instalado								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Esfuerzo de instalación	¿Qué nivel de esfuerzo se requiere para la instalación?	Contar el número de tareas de instalación automatizadas y comparar con el número de tareas definidas de la instalación	$X = A/B$ A= Número de tareas de instalación implementadas. B= Número de tareas de instalación requeridas	A se obtiene del reporte de pruebas funcionales y/o del código fuente. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican menores niveles de esfuerzo requeridos para instalar el software		Revisor de Código	
Flexibilidad de instalación	¿Cuán flexible y personalizable es la capacidad de la instalación?	Contar el número de operaciones de instalación personalizables implementadas de acuerdo a lo especificado, y comparar con el número de operaciones de instalación personalizables especificadas	$X = A/B$ A= Número de operaciones de instalación personalizables implementadas. B= Número de operaciones personalizables especificadas	A se obtiene del reporte de pruebas funcionales y/o del código fuente. B se obtiene de la especificación de requerimientos	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor capacidad de flexibilidad para instalar el software		Revisor de Código y Revisor de Requerimientos	
Métricas de coexistencia								

Métrica específica	**Propósito**	**Método de medición**	**Método de cálculo**	**Fuente de medición**	**Interpretación del valor medido**	**Valor medido**	**Responsables**	**Nota**
Capacidad de coexistencia	¿Cuán flexible es el software para compartir su entorno sin impactos adversos con otros software?	Contar el número de aplicaciones de software con las que el software puede coexistir según lo especificado, y comparar con el número de aplicaciones de software en el entorno de producción que requieran coexistencia	$X = A/B$ A= Número de aplicaciones de software con las que el software desarrollado puede coexistir según lo especificado. B= Número de aplicaciones de software en producción que requieran coexistencia	A se obtiene del reporte de pruebas funcionales y/o del reporte de pruebas no funcionales. B se obtiene de la especificación de requerimientos y/o del diseño	$0 \leq X \leq 1$ Valores de X cercanos a 1 indican mayor capacidad de coexistencia del software con otras aplicaciones de software de su entorno de producción		Revisor de Código y Revisor de Requerimientos	

Cuadro 1.9: Métricas de Portabilidad

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

A fin de llevar a cabo los procesos descritos en el Modelo de Aseguramiento de Calidad en el Desarrollo de Software Libre, es necesario plantear una estructura organizacional que se encargue de realizar las actividades de aseguramiento de calidad que conforman dicho modelo. Antes de pasar a describir la estructura mencionada se presentan algunas consideraciones que deben ser tomadas en cuenta para la conformación de dicha estructura:

- En primera instancia es necesario considerar que el modelo debe ser llevado a cabo por personas que tengan amplio conocimiento y experiencia en el desarrollo de software y en la implementación de metodologías para el desarrollo de software.
- Teniendo en cuenta el tamaño de las organizaciones que desarrollan software, las cuales para el caso venezolano son en la mayoría organizaciones pequeñas, conformadas por programadores, y en algunos casos, por personas destinadas al área de documentación del software, se plantea una estructura organizacional para el aseguramiento de calidad adaptable y flexible al tipo de organización, de modo que este modelo pueda ser aplicable tanto para pequeñas organizaciones como para grandes organizaciones. La estructura organizacional propuesta se basa en el concepto de rol, en base al cual se plantea que una persona puede cumplir varios roles a la vez en los distintos procesos de evaluación. En un caso extremo, una sola persona puede llevar a cabo todos los roles definidos en el modelo, si cuenta con aspectos y competencias requeridas para llevar a cabo los roles determinados.

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 Código (RC).
- Revisor de Pruebas Funcionales (RPF).
- Revisor de Pruebas No Funcionales (RPNF).
- Revisor de Manuales de Software (RMS).

Una de las principales actividades del equipo de aseguramiento de calidad es determinar, entre todos los miembros del equipo, las métricas que se usarán para evaluar el producto de software, las cuales se seleccionaran del conjunto de métricas propuestos en el modelo conforme sean los requerimientos funcionales y no funcionales establecidos para el software.

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.
- Emitir al equipo de desarrollo del proyecto los reportes emitidos por los miembros del equipo de aseguramiento de calidad respecto a: i) resultados de las evaluaciones que han realizado a las prácticas del proceso de desarrollo, y ii) disconformidades observadas en las prácticas de desarrollo evaluadas.
- Elaborar y emitir al equipo de desarrollo del proyecto el reporte de los resultados de la evaluación de la aplicación de software, en base a los reportes de evaluaciones de las métricas de software que emitan los integrantes del equipo de aseguramiento de calidad.

- Elaborar y emitir al equipo de desarrollo del proyecto el reporte sobre las disconformidades observadas en la evaluación de la aplicación de software, en base a los reportes de disconformidades en la aplicación de software emitidos por los miembros del equipo de aseguramiento de calidad.

A continuación en la Figura 1.1 se muestra el flujo de actividades del CEAC:

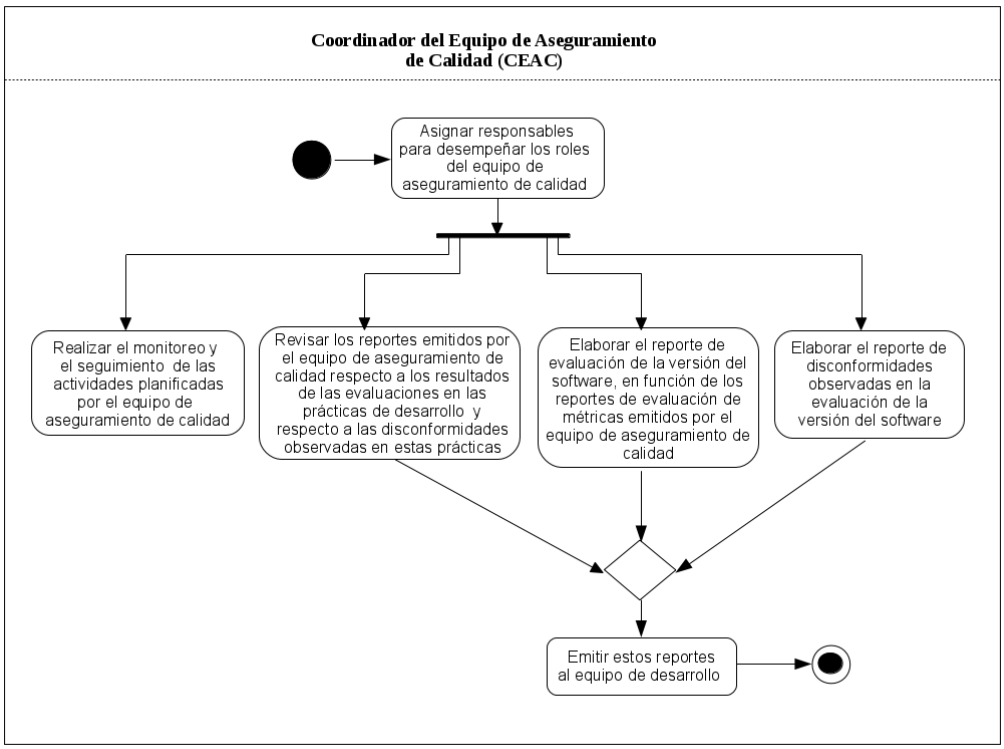

Figura 1.1: Flujo de actividades del CEAC

1.3.2. Revisor de la Conceptualización del Proyecto (RCP)

La persona que desempeñe este rol debe tener una perspectiva general del proyecto de software y de los distintos procesos, tanto del desarrollo del proyecto como del aseguramiento de calidad. Entre las principales actividades que debe realizar el RCP, se mencionan las siguientes:

- Asegurar que los objetivos de software se cumplan, por lo tanto, debe conocer bien los requerimientos exigidos por el usuario y lo especificado en el plan del proyecto de software.
- Evaluar las prácticas de desarrollo relacionadas a la Conceptualización y Administración del Proyecto, conforme al formato de evaluación propuesto en la Tabla 2, sección 1.3, y emitir esta evaluación al equipo de desarrollo del proyecto.
- Elaborar el reporte sobre las disconformidades observadas en las prácticas de desarrollo evaluadas, y emitir éste al equipo de desarrollo del proyecto.
- Repetir la evaluación para prácticas de desarrollo en las cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

A continuación en la Figura 1.2 se muestra el flujo de actividades del RCP.

1.3.3. Revisor de Requerimientos (RR)

La administración de requerimientos comprende actividades en casi todo el proceso de desarrollo de software. Las personas que desempeñan el rol de RR deben tener capacidades en relación a la elaboración de casos de uso.

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

- Asegurar el cumplimiento de los requerimientos establecidos por los usuarios.

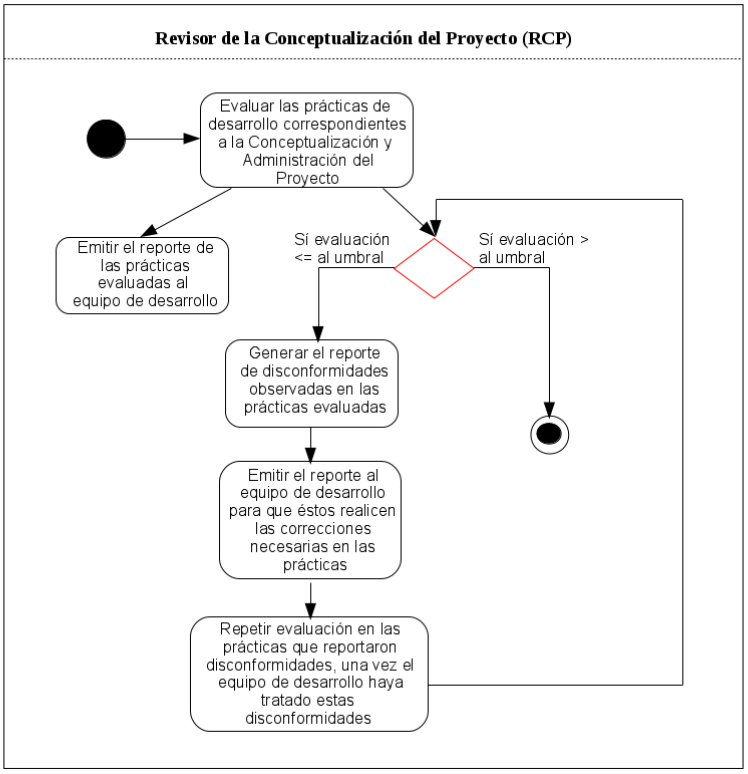

Figura 1.2: Flujo de actividades del RCP

- Evaluar las prácticas de desarrollo relacionadas a la administración de requerimientos, conforme al formato de evaluación propuesto en la Tabla 2, sección 1.3, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.
- Elaborar el reporte de disconformidades observadas en las practicas evaluadas y emitirlo al Coordinador del Equipo de Aseguramiento de Calidad.
- Evaluar las métricas de calidad de software relacionadas a la administración de requerimientos, conforme se propone en las tablas que componen la sección 1.1, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en la evaluación de las métricas de software respectivas, y emitir este reporte al Coordinador del Equipo de Aseguramiento de Calidad.
- Repetir la evaluación para prácticas de desarrollo y/o para métricas de calidad de software en las cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

A continuación en la Figura 1.3 se muestra el flujo de actividades del RCP.

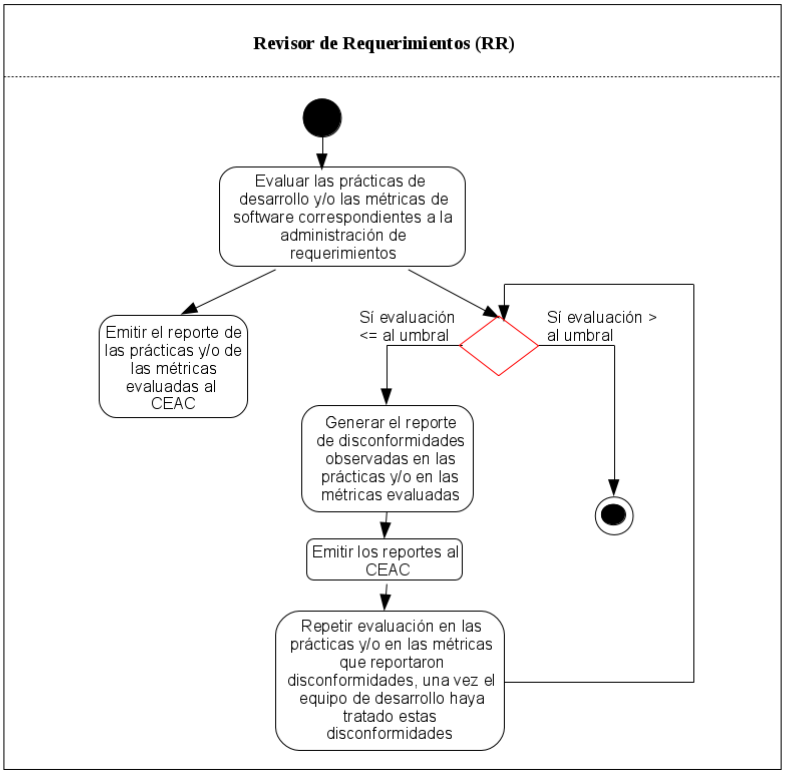

Figura 1.3: Flujo de actividades del RR

1.3.4. Revisor de la Arquitectura de Software (RAS)

Las personas que ejercen este rol deben tener capacidades en relación a la descripción 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 relacionadas al diseño de la arquitectura de software, conforme al formato de evaluación propuesto en la Tabla 2, sección 1.3, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en las practicas evaluadas y emitirlo al Coordinador del Equipo de Aseguramiento de Calidad.

- Evaluar las métricas de calidad de software relacionadas a la arquitectura de software, conforme se propone en las tablas que componen la sección 1.1, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en la evaluación de las métricas de software respectivas, y emitir este reporte al Coordinador del Equipo de Aseguramiento de Calidad.

- Repetir la evaluación para prácticas de desarrollo y/o para métricas de calidad de software en las cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

El flujo de actividades asociado a este rol es el similar al de RR, con la diferencia de que el RAS evalúa prácticas y métricas correspondientes a la arquitectura de software.

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 relacionadas al diseño de base de datos, conforme al formato de evaluación propuesto en la Tabla 2, sección 1.3, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.
- Elaborar el reporte de disconformidades observadas en las practicas evaluadas y emitirlo al Coordinador del Equipo de Aseguramiento de Calidad.
- Evaluar las métricas de calidad de software relacionadas a base de datos, conforme se propone en las tablas que componen la sección 1.1, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.
- Elaborar el reporte de disconformidades observadas en la evaluación de las métricas de software respectivas, y emitir este reporte al Coordinador del Equipo de Aseguramiento de Calidad.
- Repetir la evaluación para prácticas de desarrollo y/o para métricas de calidad de software en las cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

El flujo de actividades asociado a este rol es el similar al de RR, con la diferencia de que el RDP evalúa prácticas y métricas correspondientes al diseño y construcción de datos persistentes.

1.3.6. Revisor de Código (RC)

Los RC se encargan de la evaluación del código fuente de las aplicaciones de software. Las personas que llevan a cabo este rol debe tener capacidad y experiencia en el área de programación (codificación en diferentes lenguajes).

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

- Evaluar las prácticas de desarrollo relacionadas a la construcción de código fuente, conforme al formato de evaluación propuesto en la Tabla 2, sección 1.3, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en las practicas evaluadas y emitirlo al Coordinador del Equipo de Aseguramiento de Calidad.

- Evaluar las métricas de calidad de software relacionadas al código, conforme se propone en las tablas que componen la sección 1.1, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en la evaluación de las métricas de software respectivas, y emitir este reporte al Coordinador del Equipo de Aseguramiento de Calidad.

- Repetir la evaluación para prácticas de desarrollo y/o para métricas de calidad de software en las cuales se hallan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

El flujo de actividades asociado a este rol es el similar al de RR, con la diferencia de que el RC evalúa prácticas y métricas correspondientes al código de la aplicación de software.

1.3.7. Revisor de Pruebas Funcionales (RPF)

Las pruebas funcionales constituyen un elemento fundamental para determina si una aplicación cumple o no con los requerimientos funcionales establecidos por los usuarios.

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

- Evaluar las prácticas de desarrollo relacionadas a la elaboración y aplicación de pruebas funcionales, conforme al formato de evaluación propuesto Tabla 2, sección 1.3, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en las practicas evaluadas y emitirlo al Coordinador del Equipo de Aseguramiento de Calidad.

- Evaluar las métricas de calidad de software relacionadas a pruebas funcionales, conforme se propone en las tablas que componen la sección 1.1, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en la evaluación de las métricas de software respectivas, y emitir este reporte al Coordinador del Equipo de Aseguramiento de Calidad.

- Repetir la evaluación para prácticas de desarrollo y/o para métricas de calidad de software en las cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

El flujo de actividades asociado a este rol es el similar al de RR, con la diferencia de que el RPF evalúa prácticas y métricas correspondientes a pruebas funcionales.

1.3.8. Revisor de Pruebas No Funcionales (RPNF)

Las pruebas no funcionales son fundamentales para determinar si el software cumple o no con los requerimientos no funcionales establecidos por los usuarios. Los miembros del equipo de aseguramiento de calidad que desempeñen este rol deben tener capacidad y experiencia en la elaboración y aplicación de pruebas no funcionales, tales como: pruebas de seguridad, de rendimiento, de portabilidad, entre otras.

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

- Evaluar las prácticas de desarrollo relacionadas a la elaboración y aplicación de pruebas no funcionales, conforme al formato de evaluación propuesto Tabla 2, sección 1.3, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en las practicas evaluadas y emitirlo al Coordinador del Equipo de Aseguramiento de Calidad.

- Evaluar las métricas de calidad de software relacionadas a pruebas no funcionales, conforme se propone en las tablas que componen la sección 1.1, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en la evaluación de las métricas de software respectivas, y emitir este reporte al Coordinador del Equipo de Aseguramiento de Calidad.

- Repetir la evaluación para prácticas de desarrollo y/o para métricas de calidad de software en las cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

El flujo de actividades asociado a este rol es el similar al de RR, con la diferencia de que el RPNF evalúa prácticas y métricas correspondientes a pruebas no funcionales.

1.3.9. Revisor de Manuales de Software (RMS)

Los manuales de software constituyen una herramienta de fundamental importancia para la modificación, mejora y utilización de las aplicaciones de software. Los RMS deben tener capacidades y experiencias en la elaboración de manuales.

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

- Evaluar las prácticas de desarrollo relacionadas a la elaboración de manuales de software, conforme al formato de evaluación propuesto Tabla 2, sección 1.3, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en las practicas evaluadas y emitirlo al Coordinador del Equipo de Aseguramiento de Calidad.

- Evaluar las métricas de calidad de software relacionadas a manuales de software, conforme se propone en las tablas que componen la sección 1.1, y emitir esta evaluación al Coordinador del Equipo de Aseguramiento de Calidad.

- Elaborar el reporte de disconformidades observadas en la evaluación de las métricas de software respectivas, y emitir este reporte al Coordinador del Equipo de Aseguramiento de Calidad.
- Repetir la evaluación para prácticas de desarrollo y/o para métricas de calidad de software en las cuales se hayan reportado disconformidades en evaluaciones anteriores, a fin de verificar que tales disconformidades han sido atendidas.

El flujo de actividades asociado a este rol es el similar al de RR, con la diferencia de que el RMS evalúa prácticas y métricas correspondientes a manuales de software.

En la Figura 1.4 se muestra la intervención del equipo de aseguramiento de calidad tanto en la evaluación de las prácticas de desarrollo de software como en la evaluación de las aplicaciones de software.

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 responsables de los proyectos de desarrollo sobre las evaluaciones realizadas tanto al proceso de desarrollo como al software, así como para informar sobre disconformidades que se puedan observar durante dichas evaluaciones, a fin de que se puedan realizar las modificaciones necesarias para mejorar las prácticas de desarrollo y/o mejorar la calidad de las aplicaciones de software, a fin de cumplir con los requerimientos establecidos para las mismas. Los reportes que se deben emitir se clasifican en:

1. Resultados de evaluación por práctica del proceso de desarrollo
2. Resultados de evaluación de la aplicación de software
3. Disconformidades observadas en una práctica del proceso de desarrollo
4. Disconformidades observadas en la evaluación de la aplicación de software

La elaboración de estos reportes se describe a continuación.

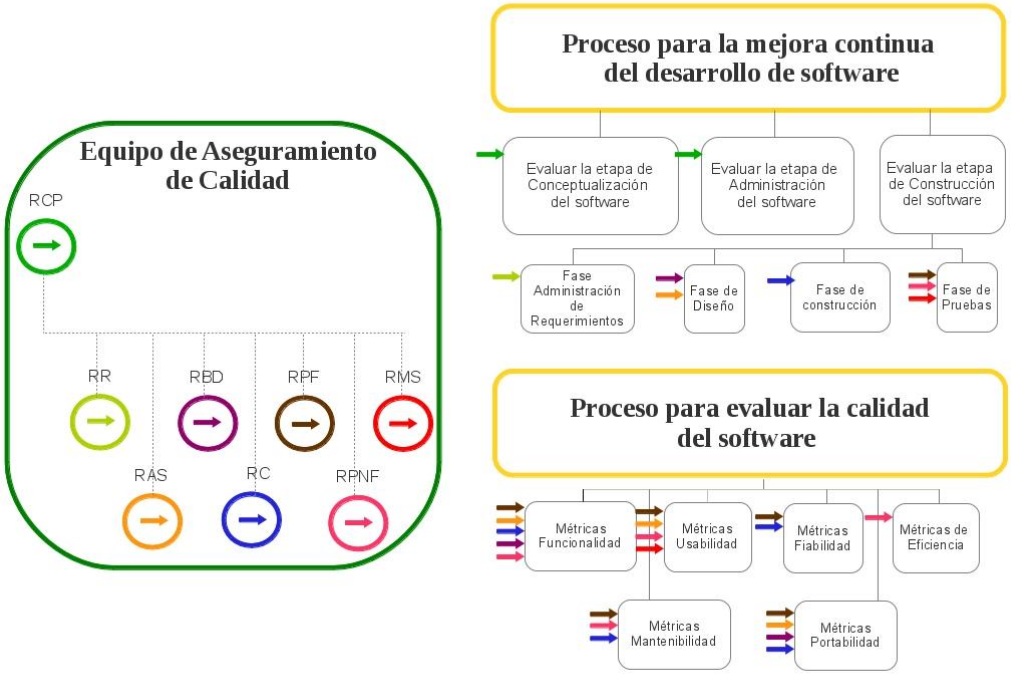

Figura 1.4: Intervención del equipo de aseguramiento de calidad en la evaluación de las prácticas de desarrollo de software y en la evaluación de las aplicaciones de software

1.4.1. Resultados de Evaluación por Práctica del Proceso de Desarrollo

En este reporte se indican las evaluaciones realizadas a las tareas que componen la práctica evaluada. El integrante del equipo de aseguramiento de calidad responsable de evaluar la práctica respectiva debe dirigir este reporte al Coordinador del Equipo de Aseguramiento de Calidad (Revisor de la Conceptualización del Proyecto), con la finalidad de que sea emitido al equipo de desarrollo del proyecto.

1.4.2. Resultados de la Evaluación de la Aplicación de Software

El reporte de evaluación de la aplicación de software contiene la evaluación de las métricas del software, y se elabora en función de los reportes que deben emitir los integrantes del equipo de aseguramiento de calidad, respecto a las evaluaciones de las métricas de software de las cuales son responsables de evaluar (ver Tablas de la sección 1.1). Este reporte debe ser elaborado por el Coordinador del Equipo de Aseguramiento de Calidad, quien se encarga de recopilar los respectivos reportes emitidos por el equipo de aseguramiento de calidad, para emitir un reporte único al equipo de desarrollo del proyecto.

1.4.3. Disconformidades Observadas en una Práctica del Proceso de Desarrollo

Este reporte se elabora en caso de observar disconformidades en la práctica evaluada, es decir, cuando tareas que conforman la práctica respectiva tengan evaluaciones menores o iguales ($\leq$) al umbral de calidad establecido para las prácticas de desarrollo. Este umbral de calidad corresponde al valor cero coma setenta y cinco (0,75).

Este tipo de reporte debe ser elaborado por el miembro del equipo de aseguramiento de calidad que evalúa la práctica respectiva, quien debe emitir dicho reporte al Coordinador del Equipo de Aseguramiento de Calidad. El objetivo de este reporte es indicar por qué el responsable de la evaluación de la práctica no está conforme con la manera en la cual se está llevando a cabo la misma.

En el Cuadro 1.10 se presenta un formato para indicar la información requerida en este tipo de reporte. En el campo "Observación de alerta" se deben

Etapa del proceso de desarrollo	**Práctica**	**Observación de alerta (disconformidad)**	**Responsable de la observación**

Cuadro 1.10: Reporte de disconformidades observadas en una práctica

indicar las razones por las cuales se considera que existe una disconformidad en la realización de la práctica.

Una vez se hayan realizado las modificaciones necesarias en la práctica, dada la disconformidad reportada, es importante que se registren dichas modificaciones en una especie de base de conocimientos, la cual permita a las organizaciones desarrolladoras de software aprovechar, para prácticas posteriores, las experiencias basadas en lecciones aprendidas durante el desarrollo de proyectos.

1.4.4. Disconformidades Observadas en la Evaluación de la Aplicación de Software

Este reporte se elabora en función de los reportes de disconformidades que deben emitir los integrantes del equipo de aseguramiento de calidad, en caso de observar disconformidades en la evaluación de la aplicación de software, es decir, cuando existan evaluaciones de métricas de software con valores menores o iguales ($\leq$) a los umbrales de calidad respectivos a dichas métricas (estos umbrales se indican seguidamente).

En el caso de las métricas de software se han definido varios umbrales de calidad, conforme a las características presentes en las métricas respectivas. En la tabla que se presenta a continuación se indican los umbrales establecidos para cada una de las métricas específicas presentadas en la sección 1.2.

Métrica	**Submétrica**	**Métrica específica**	**Umbral de calidad**	**Observaciones**
Funcionalidad	Adecuación	Adecuación funcional	0,75	
		Integridad en la implementación funcional		
		Estabilidad de la especificación funcional		
	Exactitud	Precisión de cálculos		
		Adecuación de los tipos de datos		
		Precisión		
	Interoperabilidad	Intercambiabilidad de los datos		
		Consistencia de la interfaz		
	Seguridad	Capacidad de auditar accesos		
		Capacidad para controlar los accesos		

Métrica	Submétrica	Métrica específica	Umbral de calidad	Observaciones
		Prevención de corrupción de datos		
		Cifrado de datos		
	Conformidad de funcionalidad	Cumplimiento de estándares entre sistemas		
Fiabilidad	Métricas internas de madurez	Detección de falla	0,75	
		Eliminación de fallas		
	Métricas internas de tolerancia a fallas	Fracaso al evitar		
		Evitar un funcionamiento incorrecto		
	Recuperabilidad	Restaurabilidad		
		Efectividad de la restauración		
Usabilidad	Capacidad para ser entendido	Descripción completa	0,75	
		Capacidad de demostración		
		Funciones evidentes		
		Función de entendibilidad		
	Capacidad para ser aprendido	Existencia completa de la documentación de usuario y/o ayudas		
	Capacidad para ser operado	Validación de datos de entrada		
		Capacidad del usuario de cancelar operaciones en el software		
		Capacidad del usuario de deshacer		
		Personalización		
		Accesibilidad física		
		Capacidad de supervisión del estado de las operaciones		
		Consistencia operacional		
		Claridad de los mensajes		
		Claridad de las funciones de la interfaz		
	Capacidad de atracción	Interacción atractiva		

Métrica	Submétrica	Métrica específica	Umbral de calidad	Observaciones
		Capacidad de personalización de la interfaz del usuario		
	Cumplimiento de estándares	Cumplimiento de estándares		
	Efectividad	Tareas completadas		
		Usuarios capaces de completar las tareas		
		Errores cometidos durante la realización de las tareas establecidas		
		Número de solicitudes de ayuda para llevar a cabo las tareas establecidas		
		Porcentaje de usuarios que pueden llevar a cabo las tareas sin leer el manual		
	Eficiencia	Tiempo para realizar tareas en comparación con un experto	A juzgar por los desarrolladores de la aplicación de software	Las evaluaciones positivas indican mayor facilidad de uso. Las evaluaciones con un rango de valores negativos serán conformes o aceptables según lo juzguen los desarrolladores de la aplicación de software
		Tiempo para instalar el software		
	Satisfacción	Actitud positiva ante el software	0,75	
		Satisfacción de los usuarios del software en comparación con el software anterior		
		Percepción de control con respecto al software		

Métrica	Submétrica	Métrica específica	Umbral de calidad	Observaciones
		Aporte a la productividad por el uso del software		
		Percepción de que el software apoya las tareas según sea necesario por el usuario		
		Recomendación del software a otros usuarios		
	Canales de atención al usuario	Calidad de los medios de atención al usuario		
	Documentación de usuarios	Objetivos de la documentación		
		Claridad de la documentación		
		Acceso directo		
		Terminología utilizada		
		Consistencia		
		Diseño		
		Adecuación de las fuentes		
Eficiencia	Comportamiento en el tiempo	Tiempo de respuesta	A juzgar por los usuarios y los desarrolladores de la aplicación de software	Tiempos cortos de respuesta y retorno representan buena eficiencia de la aplicación de software
		Tiempo de retorno		
		Tareas por unidad de tiempo	A juzgar por los usuarios y los desarrolladores de la aplicación de software	Mayor cantidad de tareas realizadas por encima de la unidad de tiempo establecida por desarrolladores y/o usuarios representa buena eficiencia de la aplicación de software

Métrica	Submétrica	Métrica específica	Umbral de calidad	Observaciones
	Utilización de recursos	Utilización de memoria	A juzgar por los desarrolladores de la aplicación de software	Menor tamaño de memoria utilizado por la aplicación de software para completar una tarea específica indica una mejor utilización de recursos
Mantenibilidad	Capacidad de análisis	Registro de actividades	0,75	
		Disponibilidad de las funciones de diagnóstico		
	Capacidad para ser modificado	Registro de cambios		
	Estabilidad	Impacto de cambios	A juzgar por los desarrolladores de la aplicación de software	Valores cercanos a cero indican menos impactos generados en la aplicación de software por modificaciones realizadas sobre ésta
Portabilidad	Capacidad para ser adaptado	Capacidad para adaptar las estructuras de datos	0,75	
		Capacidad para ser adaptado al entorno del hardware		
		Capacidad para ser adaptado al entorno organizacional		
		Facilidad de portabilidad para el usuario		
		Capacidad para ser adaptado al sistema operativo, al software de redes, al software de la aplicación instalada y navegadores web		

Métrica	Submétrica	Métrica específica	Umbral de calidad	Observaciones
	Facilidad para ser instalado	Esfuerzo de instalación		
		Flexibilidad de instalación		
	Coexistencia	Capacidad de coexistencia		

Cuadro 1.11: Umbrales para las métricas específicas de la evaluación de la aplicación de software

El responsable de elaborar el reporte de disconformidades observadas en la evaluación de la aplicación de software es el Coordinador del equipo de aseguramiento de calidad (Revisor de Conceptualización del Proyecto), quien se encarga de recopilar los reportes de disconformidad emitidos por el equipo de aseguramiento de calidad, para remitir un reporte único al equipo de desarrollo del proyecto. El objetivo de este reporte es indicar por qué el responsable de la evaluación de las métricas no está conforme con dicha evaluación.

Tipo de Métrica	Submétrica	Métrica específica	Observación de alerta (disconformidad)	Responsable de la observación

Cuadro 1.12: Reporte de disconformidades observadas en la evaluación de una aplicación de software

En el campo "Observación de alerta" se deben indicar las razones por las cuales se considera que existe una disconformidad en la evaluación de la métrica específica.

Al igual que en la sección 1.4.3, se recomienda para este caso registrar en una base de conocimientos las modificaciones que se realicen en el software dada las disconformidades reportadas en la evaluación del mismo.

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] M. Martín. Sistema de catalogación de métricas e indicadores con potencia de web semántica. Master's thesis, Facultad de Informática de la Universidad Nacional de La Plata, Argentina, Noviembre 2004. Grupo de Investigación y Desarrollo en Ingeniería de Software (GIDIS).