Facultad # 5

"Evaluación de la fiabilidad de los componentes de software desarrollados en el Centro de Informática Industrial"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Suleika Remedio Frometa

Tutores: M Sc. Liudmila Reyes Álvarez

Ing. Yadira Morales Álamo

Co-tutora: M Sc. Zoraida Fernández Guevara

Consultante: Dra. Marely del Rosario Cruz Felipe

La Habana

Junio del 2012

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y autorizo a la Universidad de las Ciencias Informáticas (UCI), a hacer uso de la misma en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Autora: Suleika Remedio Frometa

_____ _____

Tutora: Ing. Yadira Morales Álamo Tutora: M Sc. Liudmila Reyes Álvarez

_____

Co-tutora: M Sc. Zoraida Fernández Guevara

DATOS DE CONTACTO:

Tutores:

Nombre y Apellidos: Ing. Yadira Morales Álamo.
Institución: Universidad de las Ciencias Informáticas (UCI).
E-mail: yalamo@uci.cu.

Graduada de la UCI en el año 2008. Profesor instructor con 3 años de experiencia docente y 5 años de experiencia en la producción. Pertenece al departamento Dirección de Proyecto del Centro de Informática Industrial (CEDIN). Se desempeñó como asesora del grupo de gestión de la calidad de software en el CEDIN.

Nombre y Apellidos: M Sc. Liudmila Reyes Álvarez.
Institución: Universidad de las Ciencias Informáticas (UCI).
E-mail: lreyes@uci.cu.

Graduada de la UCI en el año 2007. Profesor asistente con 5 años de experiencia docente y 8 años de experiencia en la producción. Pertenece al departamento Integración y Despliegue del Centro de Informática Industrial. Además es Máster en Ingeniería de Software e Inteligencia Artificial de la Universidad de Málaga, España.

Nombre y Apellidos: M Sc. Zoraida Fernández Guevara.
Institución: Universidad de las Ciencias Informáticas (UCI).
E-mail: zorlis@uci.cu.

La profesora es Filóloga en Lengua Rusa, Licenciada en Educación en la especialidad de Lengua Inglesa y máster en Tecnología de los procesos educativos. Posee 28 años de experiencia en la enseñanza superior. Ha impartido asignaturas tales como Idioma Ruso e Inglés Comunicación Profesional, Formación Pedagógica, Multimedia para estudiantes no filólogos. Ha impartido un gran número de cursos de postgrados, varios de ellos preparados por ella. Ha participado en investigaciones pedagógicas, Ha presentado trabajos en eventos nacionales e internacionales. Ha publicado artículos y ponencias en memorias de eventos.

AGRADECIMIENTOS

Agradezco los esfuerzos de todos aquellos que hicieron posible que lograra todo lo que hasta hoy he alcanzado, a esta hermosa y bella Revolución por hacer posible todos mis sueños y a Fidel Castro, fundador de la Universidad de las Ciencias Informáticas.

Agradezco a mis tutoras Yadira Morales, Liudmila Reyes y Zoraida Fernández, y a la profesora Yadira Ramírez que a pesar de no ser tutora al igual que las demás me apoyó, me guió y me transmitió muchas de sus experiencias para que yo pudiera lograr mi objetivo. Gracias a todos aquellos que confiaron en mí a lo largo de mi camino como estudiante y me dieron las fuerzas para seguir adelante aún en los momentos más difíciles. Son personas maravillosas, no cambien nunca.

Agradezco, por todo el apoyo que me han brindado en todos estos años, a mis padres, en especial a mi madre Basilisa Frometa, quien ha sido mi hilo conductor.

Agradezco a Julio César por mantener siempre mi corazón lleno de amor, y darme el apoyo y la fuerza necesaria para seguir adelante.

A Pedro Osvel, por ser mi ángel de la guarda.

Agradezco a Daimi y a su familia, a Diannys, a Aliander, a Mare, en general a todos mis amigos y compañeros que me han acompañado a lo largo de estos 5 años de carrera, a todos los quiero mucho.

A todos muchas gracias.

DEDICATORIA

A mi madre, por ser el faro que ilumina mi vida,
la más hermosa prenda que ilumina mi corazón,
mi razón de ser, por estar siempre presente
y darme la confianza en mí misma
para siempre seguir adelante.

Te quiero mucho mamita.

RESUMEN

En la actualidad, el desarrollo de software implica gran cantidad de tareas y actividades en las que la perfección es una de las características más compleja de alcanzar. A pesar del desarrollo que ha alcanzado la industria del software en los últimos años, son muchos los proyectos que fracasan en el mercado y una de las causas fundamentales es la incorrecta evaluación de la calidad de software.

La presente investigación propone un procedimiento para evaluar la fiabilidad de los componentes de software que se desarrollan en el Centro de Informática Industrial (CEDIN). La fiabilidad es una de las características más importantes de la calidad de software, se entiende como el grado de precisión para que un programa realice su función prevista. Para la realización de la investigación se utilizan como guías las normas ISO/IEC 9126: Calidad de los productos y la ISO/IEC 14598: Evaluación de los productos de software.

El procedimiento está compuesto por 9 pasos: definir fases, iteraciones y tareas; describir roles y responsabilidades; definir sub-características; definir a partir de las normas utilizadas las métricas a seguir; valorar resultados; incluir el procedimiento al plan de trabajo de gestión de la calidad; describir las actividades a desarrollar en el proceso de mejora de acuerdo a la metodología a utilizar; identificar posibles errores a detectar; describir acciones correctivas que se propondrían para dar solución a los errores planteados. Cada uno de estos pasos constituye un escalón para dar cumplimiento al procedimiento.

La validación de la solución propuesta se hace en tres componentes de software. Dos de estos desarrollados en el proyecto que se conoce como: Sistema de Manejo Integral de Perforación de Pozos Petroleros (SIPP). El tercero de estos es el componente Sonido 2D-3D perteneciente a la línea de producto Realidad Virtual.

Palabras Clave: calidad de software, componente de software, evaluación de la calidad, fiabilidad

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La construcción de software está considerada como uno de los problemas de punta de este tiempo. Desde la aparición del software a mediados del siglo XX, éste ha ido ganando en complejidad. En un período de tiempo relativamente corto, el software pasó, de ser considerado un elemento secundario, a absorber una cantidad cada vez mayor de recursos económicos y acabar convirtiéndose en un factor decisivo de cara al éxito o fracaso de un sistema basado en computadora. Resulta caro desarrollar software, y los defectos de éste son particularmente costosos, tanto por los gastos de mantenimiento que generan como por la repercusión en la imagen del producto y la satisfacción de los usuarios. Hoy día, existe una clara conciencia sobre la importancia de la calidad del software, definición que será estudiada y analizada en el capítulo 1.

La necesidad de tener un producto con mayor calidad y un menor costo ha propiciado el surgimiento de una tendencia que se basa en la utilización de componentes de software. El concepto de componentes para el desarrollo de software no es un concepto nuevo. Para muchos autores simplemente es la evolución de la metodología orientada a objetos. De hecho, muchas de las características de los componentes para el desarrollo parten de la idea del diseño orientado a objetos. Pero la historia del Desarrollo de Software Basado en Componentes (DSBC) es mucho más antigua, precisamente uno, de los logros de la revolución industrial fue el desarrollo de componentes, surgiendo a partir de la necesidad de estandarizar los elementos de los productos realizados(Carleton 1994).

El DSBC busca en sus objetivos reducir el tiempo de trabajo, los costos del proyecto y el esfuerzo que requiere implementar una aplicación, incrementando el nivel de productividad de los grupos de desarrolladores y minimizando los riesgos globales sin incurrir en gastos exorbitantes. De esta forma se integran lo mejor de las tecnologías para desarrollar una aplicación de manera personalizada, a la medida de las necesidades del cliente. Hecho, éste que brinda mayor confiabilidad a las pequeñas empresas para la realización de inversiones tecnológicas.

Es importante en el DSBC contar con componentes con la calidad suficiente para asegurar que el producto final tenga un óptimo funcionamiento. Para esto es necesario tener en cuenta varias características, entre las que se encuentran la funcionalidad, la fiabilidad, la usabilidad, la eficiencia, la mantenibilidad y la portabilidad.

Una característica importante que se tiene en cuenta para medir la calidad de un componente de software es la fiabilidad, la misma trata el estudio de los fallos de productos, equipos y sistemas. La fiabilidad tiene un gran auge a nivel mundial debido a que incide directamente en el mantenimiento de los sistemas. A una mayor fiabilidad, un menor mantenimiento correctivo es necesario.

La Universidad de Ciencias Informáticas (UCI) desempeña un papel importante en la producción de software en Cuba. Actualmente existe una gran variedad y cantidad de proyectos productivos que se encuentran distribuidos en los distintos centros de desarrollo de software creados en cada facultad.

En la Facultad 5, se encuentra el CEDIN, en el cual se desarrolla software basado en componentes. Esto se fundamenta en el diseño y desarrollo de aplicaciones distribuidas basadas en componentes de software reutilizables. A raíz de esto se han desarrollado varios componentes, los cuales han sido liberados por el grupo de pruebas del departamento de integración y despliegue del mismo centro. Como ejemplos de estos componentes se tienen: Aplicaciones_v1.0, Middleware_v1.0, Seguridad, Sonido 2D-3D, DIBI de la línea gestión industrial_v1.0 y los componentes COMM3_v1.0. A cada uno de estos componentes durante el proceso de pruebas se le han detectado una gran cantidad de no conformidades, varias de ellas de tipo crítica porque constituyen fallos en el componente. Teniendo en cuenta lo anterior se hace necesario fortalecer el trabajo del rol Administrador de la Calidad con la aplicación de revisiones de evaluación a cada componente con mayor periodicidad y que permita la evaluación de cada una de las características de calidad con que debe cumplir un componente y/o un producto software. Además de que las revisiones de evaluación de la calidad realizadas a estos componentes se enmarcan solamente a los artefactos generados en cada proceso y se olvidan de que las métricas utilizadas para evaluar el producto solamente propician información de qué es lo que se va a evaluar, pero no dicen cómo evaluar la calidad de software en cuanto a las características de calidad o requisitos no funcionales (como también se les conoce) trayendo consigo que se liberen productos con los siguientes problemas:

- Los componentes de software presentan cierta inmadurez, debido a que no brindan una completa seguridad de solución inmediata en caso de ocurrir un fallo de software.
- La incertidumbre de que al ocurrir un error en el software el problema sea corregido en el tiempo requerido.
- Incompleta recuperación de la información.

- Inseguridad de los líderes de proyecto y desarrolladores de componentes de software con respecto a la fiabilidad del componente desarrollado.
- Gran cantidad de no conformidades detectadas durante la fase de pruebas internas en los componentes desarrollados por el centro, debido a las debilidades de las revisiones de evaluación realizadas durante el proceso de desarrollo del componente.

Estos problemas son aspectos negativos para el desarrollo de cualquier software debido a la insatisfacción del cliente respecto al producto esperado, porque éste no satisface sus expectativas. Aspectos que al no ser corregidos a tiempo podrían incidir en el prestigio de la universidad como desarrolladora de software.

Teniendo en cuenta como **problema científico:** ¿Cómo fortalecer las revisiones de evaluación realizadas por el Administrador de la Calidad durante el proceso de desarrollo del componente de software para que éste llegue a pruebas internas con menor cantidad de defectos?

Para solucionar esta interrogante se plantea como **objeto de estudio** la evaluación de la calidad de un componente de software.

El **objetivo general** de esta investigación es elaborar un procedimiento que permita evaluar la fiabilidad de los componentes de software desarrollados en el CEDIN.

Derivándose de éste el **campo de acción:** evaluación de la calidad de un componente de software en cuanto a su fiabilidad.

Se define como **idea a defender**:

Si se aplica en el CEDIN un procedimiento que permita evaluar la calidad de los componentes de software en cuanto a su fiabilidad durante el proceso de desarrollo del componente, se podrán disminuir las no conformidades detectadas durante la fase de pruebas internas, ejecutada por el grupo de pruebas del CEDIN.

Para darle cumplimiento al objetivo planteado, se trazan las siguientes tareas investigativas:

- Estudio y análisis del estado del arte de la calidad de componentes de software y la evaluación de la fiabilidad de los mismos, para la elaboración del marco teórico de la investigación.

- Estudio de las técnicas propuestas para la evaluación de la calidad de software, para seleccionar las técnicas a emplear en el procedimiento propuesto.
- Identificación y estudio de los principales estándares o modelos de calidad para poder determinar los que más se adecuen a la solución propuesta.
- Identificación y estudio de las distintas métricas de evaluación de la calidad de software para obtener un valor tangible de su fiabilidad.
- Definición del procedimiento para evaluar la fiabilidad de un componente de software.
- Validación del procedimiento por el método de caso de estudio en tres componentes de software para luego ser utilizado en el proceso de evaluación de la fiabilidad del grupo de gestión de la calidad del CEDIN.
- Elaboración del informe de la investigación.

La investigación se apoyará en los métodos científicos de investigación teóricos y empíricos para dar cumplimiento a las tareas expuestas anteriormente.

Dentro de los métodos teóricos se emplean:

- Analítico-sintético: para analizar y extraer los elementos más importantes que se relacionan con el objeto de estudio.
- Análisis histórico-lógico: para realizar un análisis de toda la evolución del problema que se está estudiando.
- Modelación: para la creación del procedimiento propuesto y la descripción de las diferentes alternativas, donde serán representadas las características y relaciones fundamentales del problema que se está investigando.

Dentro de los métodos empíricos se emplean:

- Observación: para adquirir la información necesaria en cualquier fase de la investigación, además de que permite ver la posible solución del problema desde diferentes puntos de vista.

- Revisión bibliográfica: para fundamentar el estudio del estado del arte mediante consultas a un conjunto de fuentes de información referidas a los temas de evaluación de la calidad y evaluación de la fiabilidad de componentes de software.
- Consulta de especialistas: para crear las habilidades necesarias a aplicar en las diferentes etapas de la investigación.
- Realización de pruebas: para verificar la validez y confiabilidad del procedimiento propuesto como resultado de la investigación.

La presente investigación está estructurada de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas y bibliografía consultada.

En el Capítulo 1: “Fundamentación Teórica”, se realiza un estudio del estado del arte donde se asimilan las definiciones de calidad de software y se investiga todo lo relacionado con la evaluación de la calidad de un componente de software y la fiabilidad del mismo, enfatizando sobre el requisito no funcional de fiabilidad de un componente de software. Se muestran definiciones y técnicas de evaluación de la calidad del mismo.

En el Capítulo 2: “Propuesta de procedimiento para evaluar la fiabilidad de software” se define y describe el procedimiento que permite al Revisor Líder y al Administrador de la Calidad evaluar la fiabilidad de los componentes de software desarrollados en el CEDIN.

En el Capítulo 3: “Validación del Procedimiento” se presentan las evidencias de validar el procedimiento propuesto en dos componentes del proyecto SIPP y en un componente de la línea de productos de Realidad Virtual del CEDIN, como resultado de la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se realiza un estudio de la calidad de software en el que se abordan las diferentes características o requisitos no funcionales (como también se les conoce) que existen para la evaluación de la calidad de software, donde se hace especial énfasis en la característica de fiabilidad. También se analizan los factores que generan las diferencias entre el estudio de la calidad de los componentes de software y el estudio de la calidad de los sistemas software en general. Además se muestra el resultado de un estudio del estado del arte en cuanto a la evaluación de la calidad de componentes de software, las técnicas de evaluación de la calidad y las definiciones y sub-características del requisito no funcional de fiabilidad.

1.2 Calidad del Software

Han sido numerosas las respuestas a la interrogante ¿qué es la calidad del software?, a continuación se muestran ejemplos más relevantes para darle respuesta a la siguiente afirmación:

“La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario” (IEEE-Sdt.610.12 1990).

“Grado con el cual el cliente o usuario percibe que el software satisface sus expectativas” (IEEE-Std.729 1983).

“Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario” (Pressman 1998).

“Conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer necesidades explícitas o implícitas” (ISO/IEC-8402 1986).

De todas las definiciones abordadas en la investigación se considera que la más completa es la que se expone en (IEEE-Std.729), debido a que recoge todos los aspectos importantes que debe cumplir un producto para que tenga una buena calidad.

Para un análisis más completo de lo que es calidad de software es necesario establecer una diferencia entre la calidad del proceso de desarrollo de software y la calidad del producto software. Este elemento es muy relevante para obtener un software con calidad, debido a que las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso, teniendo en cuenta que la calidad del producto va estar en función de la calidad del proceso de desarrollo del software. Vale destacar que sin un buen proceso de desarrollo es prácticamente imposible obtener un producto software con calidad, por lo tanto la calidad de un producto software debe ser considerada en todos sus estados de evolución.

1.2.1 Modelos de calidad del software

En un producto software la posibilidad de encontrar un conjunto de propiedades que den una indicación de su calidad está dada por la aplicación de modelos de calidad. Estos hacen que la calidad deje de ser abstracta y se convierta en algo concreto, que se puede definir, que se puede medir y, sobre todo, que se puede planificar.

Un modelo de calidad del software es un conjunto de buenas prácticas para el ciclo de vida del software, enfocado en los procesos de gestión y desarrollo de proyectos. Construir un modelo de calidad es bastante complejo y es usual que estos modelos descompongan la calidad del producto de software jerárquicamente en una serie de características y sub-características (conocidos como requisitos no funcionales), que pueden usarse como una lista de comprobación de aspectos relacionados con la calidad. Los modelos de calidad dicen QUÉ hacer, no CÓMO hacerlo. De ahí la importancia de la propuesta de este trabajo.

El estudio y análisis de la calidad de software realizado durante varios años dio lugar a los modelos de calidad del software como elemento vehicular para el análisis de la calidad de los componentes de software. Puede decirse que actualmente, los modelos de calidad son la forma estándar de describir la calidad de componentes.

Son diversas las actividades propuestas en el DSBC (Montilva 2012) en las que pueden aplicarse los modelos de calidad, entre ellas se encuentran: establecer los requisitos de calidad para la selección de un componente en base a los factores de calidad del modelo; evaluar la calidad de un componente para cada uno de los factores (características o requisitos no funcionales) de calidad del modelo; comparar la calidad

de distintos componentes respecto a los requisitos establecidos para un proceso de selección; y redactar contratos formales, donde aparezcan explícitamente las evaluaciones de calidad de los componentes que el proveedor certifica. Normalmente, los factores de calidad que aparecen en el modelo pueden utilizarse como checklist para todas aquellas cuestiones relacionadas con la calidad de los componentes.

Algunos ejemplos de modelos de calidad son los modelos de (McCall 1977), (Boehm 1978), y el modelo con un enfoque más industrial FURPS (Grady R. 1987).

Entre los modelos de calidad estudiados para la realización de este trabajo se encuentran:

- **CMMI** (CMMI-SE/SW/IPPD/SS. 2002): modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software. Con el propósito de lograr la mejora de los procesos, CMMI provee una forma de integrar los elementos funcionales de una organización, un conjunto de buenas prácticas basadas en casos de éxito examinado en organizaciones experimentadas en la mejora de procesos, ayuda para identificar objetivos y prioridades para mejorar los procesos de la organización. Dependiendo de las fortalezas y debilidades de la organización que son obtenidas mediante un método de evaluación, constituye un apoyo para que las empresas complejas en actividades productivas puedan coordinar sus actividades en la mejora de los procesos, un punto de referencia para evaluar los procesos actuales de la organización. La UCI en el año 2010-2011 logró alcanzar el nivel 2 de CMMI, facilitando estandarizar el sistema de desarrollo de software a través de procesos definidos que permitan realizar proyectos de un modo repetitivo.
- **Norma ISO/IEC 12207** (ISO/IEC-12207 1995): establece un proceso de ciclo de vida para el software que incluye procesos y actividades que se aplican desde la definición de requisitos, pasando por la adquisición y configuración de los servicios del sistema, hasta la finalización de su uso. Este estándar tiene como objetivo principal proporcionar una estructura común para que todo el personal involucrado en el desarrollo de software use un lenguaje común. Este lenguaje común se establece en forma de procesos bien definidos.
- **NC ISO 9126** (ISO/IEC-9126-1 2001): ha establecido un estándar internacional para la evaluación de la calidad de productos de software el cual fue publicado en 1992. La misma busca poder medir la calidad de un programa informático. Entendiendo por calidad: “La propiedad o conjunto de

propiedades inherentes al software que permiten determinar su valor". Para ello se propone la descomposición del atributo calidad en otros más sencillos y fáciles de medir. De esta forma se establecen los requisitos de calidad de un programa y se consigue un mayor conocimiento del programa estudiado. El estándar está dividido en cuatro partes las cuales dirigen, respectivamente, lo siguiente: modelo de calidad, métricas externas, métricas internas y calidad de las métricas en uso. En la investigación se utilizan las NC ISO/IEC 9126-1 y la NC ISO/IEC 9126-2 porque la primera trata un modelo de calidad que permite especificar y evaluar la calidad del producto software desde varias perspectivas y la segunda especifica seis características para evaluar métricas externas, que son además divididas en sub-características que se manifiestan externamente cuando el software se usa como una parte del sistema computarizado.

- **NC ISO 15504** (ISO/IEC-15504-2 2003): ha sido publicada en su versión definitiva entre los años 2003 (parte 2) y 2008 (parte 7). Está orientada a evaluar y mejorar la capacidad y madurez de los procesos. Se denomina Software Process Improvement and Capability Determination (SPICE) que su significado en español es determinación de la capacidad y mejora de los procesos del software porque se redactó como producto del proyecto SPICE promovido por el "Comité Internacional de Estándares de Ingeniería de Software y Sistemas" a través de su grupo de trabajo sobre evaluación de procesos. Hasta el momento se han publicado 7 partes que se pueden encontrar explicadas en (ISO/IEC-15504-2 2003).
- **NC ISO 14598-1** (ISO/IEC-14598 1999): ofrece una visión general, explica la relación entre su serie y el modelo de calidad de la ISO/IEC 9126, define los términos técnicos utilizados, contiene requisitos generales para la especificación y evaluación de la calidad del software, y clarifica los conceptos generales. Además, provee un marco de trabajo para evaluar la calidad de todos los tipos de productos de software y establece requisitos para métodos de medición y evaluación de los productos.

Se estudiaron estos modelos, debido a que estos proporcionan un marco de trabajo para la evaluación de la calidad, y en el caso de la ISO/IEC 14598, es el que propone la evaluación de la calidad de los productos de software.

1.3 Calidad de los componentes de software

En el DSBC el estudio de la calidad de los componentes de software juega un papel muy importante. Por ejemplo, durante los procesos de selección de los componentes es necesario conocer con detalle el comportamiento relativo a aquellos criterios a tener en cuenta para evaluar el cumplimiento de los requisitos de los sistemas en desarrollo, tanto funcionales como los no-funcionales los cuales son conocidos como características de calidad según las normas (ISO/IEC-9126-1 2001) y la (ISO/IEC-14598 1999).

Existen algunos factores que generan algunas diferencias entre el estudio de la calidad de los sistemas software respecto al estudio de la calidad de componentes de software. Según (Juan P.C 2009) entre estos factores se encuentran:

- La atomicidad: los componentes son unidades indivisibles desde el punto de vista de su gestión. Consecuentemente, podemos estudiar su calidad individualmente.
- La reusabilidad: los componentes son módulos que se reúsan e integran, en una o más aplicaciones. Ello exige un alto grado de precisión en la descripción de la calidad, especialmente en el caso de componentes Off-The-Shelf u OTS (término con el que se acostumbra a englobar los componentes COTS y FOSS, (Li 2008) y los servicios web).
- La evolución: los componentes que integran la aplicación transitan por sucesivas versiones que no se corresponden necesariamente con las versiones de los sistemas en los que se integran, especialmente en el caso de los mencionados componentes OTS. La descripción de la calidad de los componentes debe facilitar el estudio del impacto de tales evoluciones.

Para el análisis del estudio de la calidad de los componentes de software se hace necesaria la existencia de un vocabulario exhaustivo que defina con precisión las características de calidad (o requisitos no funcionales) que pueden influir en dicha calidad. Este vocabulario debe estar acompañado de indicaciones sobre los valores que dichos requisitos no funcionales pueden tomar en los componentes, y cómo se pueden obtener los mismos. Como respuesta a esta necesidad, diferentes autores y organizaciones promovieron catálogos de características de calidad. Inicialmente, estos catálogos se configuraron en forma de una lista desestructurada, cuyas características no estaban definidas con suficiente precisión (en particular sin indicaciones claras sobre los valores que podían tomar). Rápidamente, estos catálogos

evolucionaron y finalmente tomaron la forma de modelos, estándares y/o normas de calidad (Juan P.C 2009). De ahí que el núcleo en el estudio de la calidad de componentes de software sea el establecimiento de métricas y/o indicadores que permitan obtener un valor cuantitativo de cada una de las características de calidad a las que se hace referencia en las normas: (ISO/IEC-14598 1999) y la (ISO/IEC-9126-1 2001). Ambas normas constituyen el núcleo de este trabajo porque son las que explican de manera más detallada "qué hacer" para evaluar cada una de las características de calidad de software. Por lo tanto se hace menos complejo para el Administrador de la Calidad llevar a cabo el "cómo hacer" la evaluación de la característica de fiabilidad, siendo esta el núcleo del objetivo de la investigación.

1.3.1 Investigaciones sobre fiabilidad de componentes de software

La capacidad de predecir la fiabilidad de un sistema de software al inicio de su desarrollo puede ayudar a mejorar la calidad de software del mismo. La fiabilidad de software es un factor clave en la seguridad de sistemas como los sistemas de salud y los controladores de tráfico, es también uno de los atributos de calidad más importantes en los sistemas integrados como pueden ser redes de sensores que producen información para los procesos de negocio por lo tanto, las decisiones de diseño que tienen un gran impacto en la fiabilidad de un sistema software, la arquitectura y los componentes, tienen que ser evaluados a fondo.

A continuación se exponen algunos ejemplos de fiabilidad de componentes de software:

El trabajo (Cheung 2008) se centra en los principales desafíos del desarrollo de un software de acuerdo a la fiabilidad de sus componentes, teniendo en cuenta modelos y técnicas de análisis que permiten evaluar la fiabilidad de los componentes. En un principio cuando se hablaba de predicción de fiabilidad para el desarrollo de sistemas software complejos, se puede decir que los mismos eran críticos. Sin embargo, los primeros esfuerzos en esta área han permitido asumir algún grado de conocimientos en cuanto a componentes individuales y si de funcionamiento se trata. Se ha presentado un marco para evaluar los componentes en cuanto a su fiabilidad, basado principalmente en cuatro fases de modelado. Los resultados de evaluación indican principalmente que el marco de evaluación proporciona una significativa predicción de fiabilidad en el contexto de las primeras etapas del desarrollo de software. La investigación de este tema sigue en curso, las técnicas y métodos se siguen perfeccionando y siguen siendo un reto a nivel de los sistemas software.

Otra investigación importante que trata las técnicas de fiabilidad del software es precisamente (Roshandel 2006), aquí estas técnicas tienen como objetivo reducir o eliminar fallas en los sistemas software. La fiabilidad en los sistemas software se mide durante o después de la implementación del sistema, sin embargo, esta metodología tiene como ideología “Las cosas correctas”, resolverlas en un principio del ciclo de desarrollo del software evitando los gastos exagerados que puedan ocasionar las fallas y el mantenimiento que haya que hacerle al sistema. Este trabajo propone calcular la estimación de la fiabilidad de un componente de software en cuanto a su arquitectura. Con esta investigación se intentó cerrar la brecha abierta que ha existido durante varios años entre modelado, análisis y medición de la fiabilidad de software, cuantificando los defectos y el costo del comportamiento de cómo se comporta un fallo en un componente de software. Para ello se utilizaron varios métodos, como ejemplos se pueden ver los métodos de simulación de estados y los métodos de seguimiento de la afirmación de especificación del módulo. A fin de estimar el nivel de fiabilidad del sistema se investiga un modelo basado en componentes de software que resuelva el problema de manera eficiente de lo contrario el sistema propuesto se vuelve intratable o inútil para evaluar la fiabilidad en sistemas grandes. La validación de la propuesta se realizó en sistemas en tiempo real.

El trabajo (Hamlet 2001) trata una teoría fundamental de la fiabilidad del software basado en componentes, la teoría propone cómo los desarrolladores prueban y diseñan los componentes para luego ser utilizados por los diseñadores del sistema para calcular la confiabilidad del mismo. Se describe cómo realizar mediciones a componentes independientes, en principio la teoría resuelve el problema central de una evaluación que es: un desarrollador de componentes cómo puede saber si el componente que está desarrollando es fiable, cómo puede asegurar al cliente que cumple con las especificaciones que éste declaró, sin ningún problema, para ellos es necesario que el desarrollador certifique o valide la teoría que se está probando, de esta manera se puede certificar un componente para ser liberado y ser usado en el sistema software que lo requiera. Para ello el desarrollador utilizó una base de datos para identificar el conjunto de subdominios apropiados para llevar a cabo la medición del componente de software.

La investigación (Ovaska 2011) no trata la evaluación de la fiabilidad durante el diseño y la implementación sino que proporciona un enfoque coherente mediante la combinación de ambas, midiendo valores con las estimaciones heurísticas con el fin de facilitar el proceso de evaluación. El uso de este enfoque de fiabilidad para la evaluación desarrollado con la cadena de herramientas de apoyo se ilustra con un caso de estudio. La contribución es un enfoque que integra la estimación de la heurística de la

fiabilidad, basadas en modelos de predicción y en actividades de medición de la fiabilidad a nivel de componentes para apoyar el incremento y el desarrollo iterativo de información confiable basada en sistemas de componentes de software. Como resultado de este trabajo se tiene que es posible combinar los métodos para un modelo de arquitectura y utilizarlos en el nivel de predicción del sistema de fiabilidad. La fiabilidad y la cadena de herramientas de apoyo se han desarrollado y evaluado de forma incremental en diferentes investigaciones y proyectos piloto como en los que se basó este trabajo.

Atendiendo a los resultados expuestos anteriormente respecto a la fiabilidad de componentes de software es importante destacar que esta investigación se va a centrar en las revisiones de evaluación al proceso de desarrollo del componente. De todos los estudios antes expuestos, aportaron a esta investigación (Roshandel 2006) y (Cheung 2008) debido a que ambos parten de la idea de resolver los problemas detectados con respecto a la fiabilidad de un componente durante el ciclo de desarrollo del mismo pero enfocado al resultado, es decir al componente que se obtiene. Aspecto que evita los gastos exagerados que puedan ocasionar las fallas y/o no conformidades detectadas en la fase de pruebas internas del componente. El análisis de los trabajos presentados en (Roshandel 2006) y (Ovaska 2011) permitieron que se profundizara en toda la teoría que rodea la fiabilidad de componentes de software y los elementos a tener en cuenta cuando se hace una estimación de la fiabilidad de un componente, como por ejemplo: la definición del proceso de evaluación.

1.3.2 Características de calidad de software

A continuación se explican brevemente cada una de las características que hacen a un software de calidad según la norma (ISO/IEC-9126-1 2001).

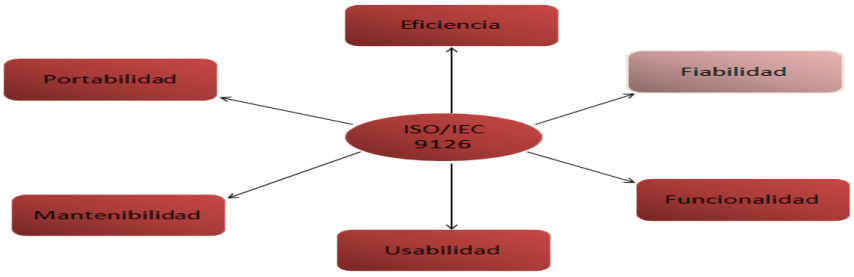

Fig. 1: Características de Calidad de Software.

- **Funcionalidad**

Es la capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas implícitas cuándo el software se usa bajo las condiciones especificadas.

- **Usabilidad**

Capacidad del producto software de ser comprendido, aprendido, utilizado y de ser atractivo para el usuario, cuando se utilice bajo las condiciones especificadas.

- **Eficiencia**

Capacidad del producto software para proporcionar una ejecución o desempeño apropiado, en relación con la cantidad de recursos usados bajo condiciones establecidas.

- **Mantenibilidad**

Capacidad del producto software de ser modificado. Las modificaciones pueden incluir las correcciones, mejoras o adaptaciones del software a cambios en el ambiente, así como en los requisitos y las especificaciones funcionales.

- **Portabilidad**

Capacidad del producto software de ser transferido de un ambiente a otro.

- **Fiabilidad**

La capacidad del producto software para mantener un nivel de ejecución especificado cuando se usa bajo las condiciones especificadas.

Es de interés de este trabajo solo el requisito no funcional de fiabilidad, por lo tanto en la próxima sección se expondrá un análisis del estudio del estado del arte de la fiabilidad del software.

1.4 Fiabilidad

Son muy diversas las definiciones de fiabilidad que aparecen en la literatura actual. Por lo tanto se decide en la investigación realizar un estudio de cómo ha evolucionado esta definición:

Según estándar IEEE Software Quality Metrics (IEEE-Sdt.610.12 1990) una definición ampliamente aceptada de fiabilidad del software es: "la probabilidad de que un programa realice su objetivo

satisfactoriamente (sin fallos) en un determinado período de tiempo y en un entorno concreto (denominado perfil operacional)".

Según (McCall 1977) el término fiabilidad se define “hasta dónde puede quedarse un programa que lleve a cabo su función pretendida con la exactitud solicitada”.

Según (CORPORACION-UNIVERSITARIA-REMINGTON 2010) la fiabilidad del software se define en términos estadísticos como “la probabilidad de operación libre de fallos de un programa de computadora”.

Según (Hamlet 2001) la fiabilidad (reliability) de un sistema es una medida de su conformidad con una especificación autorizada de su comportamiento. La especificación de la misma debería ser completa, consistente, comprensible y no ambigua.

(Sommerville 2005) define la fiabilidad del software como: “la probabilidad de que, durante un período de tiempo, el sistema funcione correctamente tal y como espera el usuario. La probabilidad de operación libre de fallos de un programa de ordenador durante un tiempo especificado y en un entorno específico”.

La norma (ISO/IEC-9126-1 2001) la define como: “Un conjunto de atributos que tienen que ver con la capacidad del software de mantener su nivel de rendimiento bajo condiciones dadas por un período de tiempo definido”.

Partiendo de la idea, que esta investigación debe evaluar la fiabilidad de componentes de software durante el proceso de desarrollo del mismo. Se decide definir a la fiabilidad como una característica de calidad que puede ser evaluada durante las revisiones de evaluación realizadas por el rol de Administrador de la Calidad durante el proceso de desarrollo del componente, permitiendo que este cumpla con el objetivo para el cual fue confeccionado, sin la ocurrencia de fallos en un determinado período de tiempo, en el entorno concreto y sistema para el cual fue desarrollado el componente.

1.4.1 Fiabilidad de software

La fiabilidad de un software, sin lugar a dudas es un elemento importante de su calidad en general. Si un software falla frecuente y repetidamente en su funcionamiento, no importa si el resto de los factores de calidad son aceptables.

La fiabilidad del software, a diferencia de otros factores de calidad, puede ser medida o estimada mediante datos históricos o de desarrollo. La fiabilidad del software se define en términos estadísticos

como la probabilidad de operación de un sistema libre de fallos en un entorno determinado y durante un tiempo específico (Nicanor 2003).

Por ejemplo, un programa X puede tener una fiabilidad estimada de 0,96 durante un intervalo de proceso de ocho horas. En otras palabras, si se fuera a ejecutar el programa X 100 veces, necesitando ocho horas de tiempo de proceso (tiempo de ejecución), lo probable es que funcione correctamente (sin fallos) 96 de cada100 veces.

Siempre que se habla de fiabilidad del software, surge la interrogante: ¿Qué se entiende por el término fallo?

En el contexto de cualquier discusión sobre calidad y fiabilidad del software, el fallo es cualquier falta de concordancia con los requisitos del software. Incluso en esta definición existen grados. Los fallos pueden ser simplemente desconcertantes o ser catastróficos. Puede que un fallo sea corregido en segundos mientras que otro lleve semanas o incluso meses. Para complicar más la situación, la corrección de un fallo puede llevar a la introducción de otros errores que, finalmente, lleven a más fallos.

La fiabilidad de software ha sido ampliamente estudiada por investigadores de universidades de todo el mundo, que responden a proyectos de investigación de los cuales se recibe una remuneración si se brindan soluciones que garanticen o comprueben, de alguna forma la fiabilidad o no de un software y/o componente de software.

1.4.2 Resultados alcanzados en la fiabilidad de software

El creciente tamaño y complejidad de las aplicaciones software, la investigación en el ámbito de la fiabilidad del software basado en la arquitectura ha ganado gran importancia. Hoy en día, la reutilización del software ha atraído la atención de todos en la industria del software. La mayoría de los métodos utilizados en la evaluación no satisfacen las necesidades, debido a que estos métodos se idearon teniendo como base un solo producto. Además, aunque existen muchos enfoques para la evaluación del software, la aplicación de estos suele ser costosa y consume mucho tiempo. Esto indica que la reutilización no es un problema para la evaluación de un producto. La sociedad moderna depende en gran medida de los sistemas software complejos para las actividades cotidianas. La fiabilidad de estos sistemas se ha convertido en una característica crítica que determina qué productos serán exitosos y cuáles serán ampliamente aceptados por el cliente.

Algunos ejemplos relevantes de investigaciones que aportan a la evaluación de la fiabilidad de un software, se muestran a continuación:

En el trabajo (Reza 2011) se presenta un modelo de red bayesiana para evaluar la fiabilidad de la línea de montaje llamada “Arquitectura”. Para la evaluación no sólo se ha tenido en cuenta el papel de los elementos en la fiabilidad del sistema, sino también se ha considerado la relación entre los elementos de evaluación. La evaluación se basa en el conjunto de cambios de la línea de productos “Arquitectura”, el método presentado implica no sólo los elementos, sino también las relaciones en la evaluación de la fiabilidad, esto hace que se desempeñe un papel importante en los entornos dinámicos y sistemas móviles. De esta manera se resume el proceso de reutilización de los productos los cuales se evalúan permitiendo ahorrar tiempo y costos. La principal ventaja es que utiliza productos reutilizables, incorporados en los conjuntos de cambios, y la evaluación de la superficie. El enfoque propuesto se basa en "la regla de Bayes".

En el caso del estudio realizado en (Roshandel 2007) este presenta un enfoque para el modelado de la fiabilidad en sistemas de software a nivel arquitectónico. Las redes bayesianas dinámicas son utilizadas para construir un modelo de fiabilidad que se basa en los modelos estándar de arquitectura de software, y no requiere de la implementación a nivel de artefactos. Los valores de confiabilidad obtenidos a través de este enfoque pueden ayudar al arquitecto en la evaluación. El enfoque se evaluó utilizando la sensibilidad y la incertidumbre. A pesar de la madurez de las técnicas de fiabilidad del software, la predicción de la fiabilidad de sistemas de software antes de su implementación no ha recibido la atención adecuada en el pasado. Este trabajo demuestra, con evidencias, que la evaluación a tiempo al software puede liberar productos de mejor calidad y más rentables. Teniendo en cuenta las incertidumbres asociadas a los sistemas software a principios del proceso de desarrollo, se evalúan modelos apropiados de fiabilidad que deben ser capaces de acomodar las incertidumbres y producir resultados significativos. En este caso el enfoque para la predicción de la fiabilidad aprovecha las redes bayesianas dinámicas y calcula la fiabilidad general teniendo en cuenta componentes individuales, así como sus complejas interacciones.

El resultado de la investigación descrita en (Dill 2009) propicia una mayor flexibilidad en la descripción de la práctica de sistemas. En este modelo, existen componentes en el sistema donde cada componente adquiere valores que se encuentran entre (0,1,…,n). Aunque el modelo tiene varias aplicaciones prácticas, los métodos existentes para la informática ya sea la fiabilidad exacta, o aproximada de estos

sistemas son computacionalmente ineficientes, y se limita a sistemas muy pequeños. En este trabajo, se propone un método eficiente, y un algoritmo para calcular detalladamente la fiabilidad exacta de multi-estado-fuera-de-sistemas. El método se basa en las probabilidades condicionales, y es aplicable a todos los casos de multi-estado-fuera de los sistemas de: 1) constante, 2) disminuyente, 3) aumentado, y 4) valores no monótonos. El algoritmo propuesto es rápido y robusto. La fiabilidad de este algoritmo se puede calcular en un corto tiempo. Por ejemplo, la fiabilidad exacta de un sistema con 500 componentes, con 200 estados posibles se puede calcular en menos de un segundo. Se considera que ilustran la eficacia y la eficiencia del método propuesto. Además de que el algoritmo detalla antecedentes teóricos, y ofrece la lista completa del código MATLAB utilizado en los cálculos.

El artículo (Carrington 2004) propone un marco conceptual para la evaluación de la fiabilidad de componentes de software que incorpora la ejecución de casos de prueba y la producción de la evaluación. La determinación de un perfil operativo y la evaluación de la prueba de salida son dos problemas difíciles e importantes que se abordan en esta investigación. La determinación de un perfil operacional es difícil porque requiere anticipar el uso futuro de un componente. Un resultado esperado es necesario en cada caso de prueba para evaluar el resultado de la prueba utilizando para ellos la base de datos *Oracle*, donde se generan los resultados esperados. El marco combina las pruebas estadísticas y las pruebas realizadas como la comprobación del auto-versión de las implementaciones. La investigación ilustra a través de dos ejemplos claros que fueron seleccionados la identificación de los problemas que deben ser dirigidos a proporcionar soporte a las herramientas utilizadas.

Otro de los estudios interesantes fue el descrito en (Vergilio 2010) el cual emplea modelos que permiten adaptarse mejor a la curva de fiabilidad cuando se comparan con otra forma tradicional. En trabajos anteriores los investigadores llevaron a cabo experimentos con modelos basados en el tiempo. En esta investigación se evalúa un enfoque, llamado Programación Genética y Fomento (teniendo como siglas en inglés *GPB*), que utiliza técnicas para mejorar el rendimiento de la *GP*. Este enfoque presenta mejores resultados que la clásica *GP*, pero requiere diez veces el número de ejecuciones. Por lo tanto, se introduce en este trabajo un nuevo enfoque basado en la *GP*. Por lo tanto, es una técnica excelente y menos costosa para modelar la fiabilidad del software.

Otra exploración destacada es la desarrollada en (Rantanen 2007) que representa un enfoque de segunda generación en el análisis de la fiabilidad humana (conocida por sus siglas en inglés *HRA*). Aunque vale

señalar que el método utilizado es muy tedioso para aplicar manualmente y no está aún en uso generalizado, debido a que no ha sido muy probado. Para permitir una evaluación rápida y sistemática del método *CREAM*, se ha desarrollado una herramienta de software. En la investigación se muestran los resultados obtenidos al evaluar los métodos en la herramienta. Además se evidencia la simplicidad del software desarrollado permitiendo a los principiantes analizar fallos que puedan ocurrir en el software.

En el caso del trabajo (Goaeva-Popstojanova 2011) trata el impacto de las fallas de implementación de acuerdo a su fiabilidad. Se estudia esta afirmación en un software de código abierto conocido: el Eclipse. En el mismo se investiga el fracaso de la tendencia de los componentes comunes (son los que se vuelven a utilizar en todos los productos). La alta reutilización de los componentes de variación (reutilizados en cinco o seis productos) y la baja reutilización de los componentes de variación (reutilizado en uno o dos productos) es decir, cómo Eclipse evoluciona. También se estudia hasta qué punto los componentes comunes y su variación cambia con el tiempo tanto en términos de la adición de nuevos ficheros como la modificación de archivos existentes. Entre los resultados cuantitativos principales de la investigación referente a la minería, el análisis de error de *Eclipse* y los repositorios de liberación, muestran que el producto línea de evolución, presenta menos fallas graves en los componentes que se aplican comúnmente y que estos componentes también exhiben menos cambios con el tiempo. Estos resultados fueron aproximadamente como se esperaba. Sin embargo, contrariamente a lo esperado, la aplicación de las variaciones de los componentes, incluso cuando se reutilizan en cinco o más productos, continúan desarrollándose con bastante rapidez. En los resultados obtenidos se muestran que el número de fallos graves en los componentes de variación no muestra un patrón uniforme de disminución en el tiempo.

Los resultados del estudio elaborado en (Wang 2007) presentan una clase muy importante de modelos de software de fiabilidad y es ampliamente utilizado en la ingeniería de software. La estimación no paramétrica del proceso no homogéneo de Poisson (*NPCPs* como lo indican sus siglas del inglés) es caracterizada por sus funciones de intensidad. En la literatura se suele suponer que las formas eficaces de las funciones de intensidad son conocidas y solamente algunos parámetros son desconocidos. Los métodos estadísticos paramétricos pueden aplicarse entonces para estimar o para probar la fiabilidad de los modelos desconocidos. Sin embargo, en situaciones realistas a menudo es el caso de que la forma funcional de la intensidad de la falla no es muy conocida o es completamente desconocida. En este caso, se debe usar una función no paramétrica: los métodos de estimación. Las técnicas no paramétricas no requieren ninguna hipótesis preliminar sobre los modelos de software. Los actuales métodos no

paramétricos en la estadística no suelen ser aplicables a los datos de fiabilidad del software. En este trabajo se construye un método no paramétrico para estimar la función de la intensidad de fracaso del modelo *NPCP*, tomando en consideración los datos particulares del software cuando este fracasa.

La fiabilidad en sistemas de información a internet según (Nicanor 2003) se evalúa teniendo en cuenta dos entornos, el entorno real y el entorno ideal. En el entorno ideal se evalúa y modela el comportamiento del sistema bajo condiciones ideales y en el entorno real se evalúa el comportamiento real del mismo. Al concluir la evaluación que se realiza en ambos entornos se comparan los resultados y se saca una conclusión de que tan fiable es el sistema real con respecto al comportamiento que debería tener bajo condiciones ideales. El proceso de evaluación que se realiza en ambos entornos se componen por una serie de pasos lo cuales se describen a continuación:

- Determinación de las condiciones iníciales: en este paso se definen los valores de cada variable que se va a utilizar en el proceso de evaluación.
- Selección del atributo de la calidad y de sus métricas de software: en este se selecciona la fiabilidad como atributo de la calidad y se utilizan para evaluar dicho atributo las métricas densidad de defectos y media de ocurrencia de fallos.
- Proceso de Medición: en este paso se seleccionan los componentes a evaluar, se miden las características de los componentes con las métricas definidas en el paso anterior, se identifican las mediciones anómalas y se identifican los componentes anómalos.
- Evaluación de los resultados y selección del modelo: en este paso se evalúan los resultados obtenidos en el paso anterior y se selecciona el modelo a utilizar.
- Estimación de los parámetros del modelo: en este paso se escoge el tipo de ley de probabilidad y se evalúan los parámetros contenidos en la misma.
- Sustitución de los parámetros obtenidos: en este paso se sustituyen los parámetros obtenidos en el paso anterior en el modelo.
- Validación del modelo: en este paso se realiza la validación del modelo obtenido utilizando una curva de Weibull ya que si el comportamiento del modelo obtenido sigue una curva de Weibull

permite afirmar que las métricas escogidas son las adecuadas para representar el comportamiento de la fiabilidad.

- Realización de las predicciones de la fiabilidad: en este paso se realizan las predicciones de calidad teniendo en cuenta el modelo validado en el paso anterior.

El método de aplicación, descrito anteriormente, tiene como ventaja fundamental que engloba la fiabilidad desde una perspectiva genérica, facilitando que la evaluación sea sencilla y con resultados bastante exactos. Sin embargo, al evaluar la fiabilidad de manera general solamente tiene en cuenta los fallos encontrados en el sistema lo que puede provocar que en un futuro el mismo falle por motivos presentes en otras sub-características que no se tuvieron en cuenta en el momento de evaluar la fiabilidad como son: la madurez y la recuperabilidad.

Según lo descrito en (Llarena 2008), la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de San Juan está realizando propuestas de cursos bajo la modalidad no presencial y es su preocupación concretarlas en un marco de calidad. El motivo de análisis está centrado en determinar aspectos, características e instrumentos de evaluación de las actividades del proceso educativo que deben ser valorados. Para crear mecanismos que aseguren la calidad de estos procesos, surge en la universidad un proyecto que responde a la problemática de: ¿Cómo se puede determinar la fiabilidad de los instrumentos utilizados? En el contexto de esta evaluación se estudiaron diferentes métodos para lograr determinar cuán fiables eran los instrumentos utilizados para medir el proceso educativo. Para una profundización de estos métodos se recomienda acceder a (Llarena 2008).

Según la propuesta presentada en (2010) se pretende apoyar la calidad del servicio en los sistemas intensivos para lo cual se utilizaron dos variables para evaluar la fiabilidad, el tiempo entre fallos (*TBF* siglas en inglés) y el número de fallos observados por unidad de tiempo (*FC* siglas en inglés), evaluados ambos parámetros en el método de Scheneidewind el cual plantea:

M (i) = a/b (1-exp (-bi))

a: número de fallos observados por unidad de tiempo.

b: tiempo medio entre fallos.

Después de las pruebas, con un nivel de confianza del 95% deben quedar menos de 10 errores residuales en el sistema con un impacto en la caída del sistema, pero no quedarán errores críticos que afecten a la integridad del resultado obtenido.

Basándose en un análisis de Pareto, el equipo técnico de (Fernando 2009) decidió analizar la fiabilidad de la bomba de GNL 510-P-01C. El equipo pensaba que el sistema de la bomba tenía una baja fiabilidad porque la condición del proceso había cambiado en comparación con las condiciones del diseño original. Se realizó una investigación en la base de datos del *Computerized Maintenance Management System* (CMMS) conocido en español como sistema informatizado de gestión del mantenimiento para la bomba de GNL 510-P-01C que indicó que el modo de fallo más frecuente estaba asociado con el fallo mecánico de una junta mecánica.

A partir de los datos recogidos para esta bomba, en relación con la base de datos, se seleccionó el software para aplicaciones de fiabilidad, por su capacidad para llevar a cabo un análisis de Weibull. La fórmula empleada para el cálculo de la fiabilidad es la siguiente:

$$R_{t=e-\left(\frac{t}{n}\right)\beta, t>0}$$

R (t) = valor de la fiabilidad. Debe tomar valores entre 0 y 1.

t = antigüedad del fallo. Esta dado en horas, ciclos.

n = parámetro de escala. Esta dado en horas, ciclos.

β = parámetro de forma (β <1; β =1; β >1).

Una de las ventajas de aplicar el análisis de Weibull es el hecho de que proporciona un perfil de modelización flexible que cubre los patrones de fallo relativo a juventud, aleatoriedad y desgaste. Después de los cálculos realizados se arrojó como resultado que el tiempo medio entre fallos (*MTBF* como indican sus siglas en inglés) de la junta mecánica es de 8.518 horas, lo que indica que el 50% de las juntas mecánicas de las bombas fallan antes de alcanzar este número de horas de trabajo, y el 50% falla pasado ese momento. El análisis ha permitido al cliente la actualización del sistema de bombas, mejorando la junta mecánica. La fiabilidad de la modificación se controla mediante un análisis regular de los datos con el método de Weibull, que permite determinar la mejora de la fiabilidad gracias a la ampliación del *MTBF* por encima de la base de referencia establecida originalmente (Fernando 2009).

Como se aprecia en los ejemplos descritos anteriormente, la fiabilidad de software constituye un factor esencial cuando se estudia la calidad de componentes de software, lo que demuestra la necesidad de este trabajo para el CEDIN y la repercusión del mismo en el DSBC (Montilva 2012). En los ejemplos anteriores solo se tiene en cuenta la sub-característica de tolerancia a fallos y sin embargo se deja de lado el estudio y análisis de las restantes sub-características de calidad de la fiabilidad. Elementos que en nuestra investigación es incluido intencionalmente, esclareciendo que fiabilidad no es solo tolerancia ante fallos.

Dentro de los atributos o sub-características (según la norma (ISO/IEC-9126-1 2001)) de la fiabilidad de software se encuentran:

Fig. 2: Sub-características o atributos específicos de la fiabilidad.

- **Madurez:** Capacidad del producto software de evitar un fallo total como resultado de haberse producido un fallo del software.
- **Tolerancia ante fallos:** Capacidad del producto de software de mantener un nivel de ejecución o desempeño especificado en caso de fallos del software o de infracción de su interface especificada.
- **Recuperabilidad:** Capacidad del producto de software de restablecer un nivel de ejecución especificado y recuperar los datos directamente afectados en caso de fallo total.

1.5 Técnicas de evaluación de la calidad del software

Durante la evaluación de la calidad de un software se debe tener en cuenta que el nivel de propensión de faltas de un sistema y/o componente de software afecta siempre a varias de las características o requisitos no funcionales de calidad. En particular la fiabilidad y la funcionalidad constituyen los factores más afectados. Sin embargo, no se evidencia una relación bien establecida entre las faltas y los fallos cuando a calidad se refiere. Todas las faltas de un producto software no se manifiestan como fallos. Las

faltas se convierten en fallos cuando el usuario de un sistema software nota un comportamiento erróneo. Para que un sistema software alcance un nivel alto de calidad se requiere que el número de fallos sea el menor posible. Pero para mantener los fallos a niveles mínimos las faltas necesariamente deben también estar en niveles mínimos. Para entender mejor esta descripción se muestran los siguientes significados:

Error: acción humana que produce una falta.

Falta: algo que está mal en un producto (modelo, código, documento, etc.).

Fallo: manifestación de una falta.

Defecto: error, falta o fallo.

El arma fundamental para el control de la calidad de un software lo constituyen las distintas técnicas de evaluación. Estas son las principales estrategias para detectar faltas y fallos.

Actualmente, según el nivel 3 de CMMI (CMMI-SE/SW/IPPD/SS. 2002) y la (IEEE-Std.729 1983), se pueden distinguir dos tipos de evaluaciones durante el proceso de desarrollo de software: verificaciones y validaciones. Éstas se definen como:

- Verificación: proceso de determinar si los productos de una determinada fase del desarrollo de software cumplen o no los requisitos establecidos durante la fase anterior.
- Validación: proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de las necesidades del cliente.

De ahí que la verificación ayude a comprobar si se ha construido el producto correctamente, mientras que la validación ayuda a comprobar si se ha construido el producto correcto. En otras palabras, la verificación tiene que ver típicamente con errores de los desarrolladores que no han transformado bien un producto del desarrollo en otro. Mientras que la validación tiene que ver con errores de los desarrolladores al malinterpretar las necesidades del cliente. Así la única persona que puede validar el software, ya sea durante su desarrollo como una vez finalizado, es el cliente, ya que será quien puede detectar si hubo o no errores en la interpretación de sus necesidades.

Tanto para la realización de verificaciones como de validaciones se pueden utilizar distintos tipos de técnicas. En general, estas técnicas se agrupan en dos categorías:

- Técnicas de evaluación estáticas: buscan faltas sobre el sistema en reposo. Es decir, estudian los distintos modelos que componen el sistema software buscando posibles faltas en los mismos. Así pues, estas técnicas se pueden aplicar, tanto a requisitos como a modelos de análisis, diseño y código.
- Técnicas de evaluación dinámicas: generan entradas al sistema con el objetivo de detectar fallos, al ejecutar dicho sistema sobre esas entradas. Se pone el sistema a funcionar buscando posibles incongruencias entre la salida esperada y la salida real. La aplicación de técnica dinámicas es también conocida como pruebas del software o testing y se aplican generalmente sobre código que es, hoy por hoy, el único producto ejecutable del desarrollo.

Ambas técnicas han sido aplicadas en el procedimiento que se describirá, en el capítulo 2 de esta investigación. La primera de ellas se aplica al componente en reposo tanto en la fase de requisitos como en fases posteriores. La segunda es la técnica que todo Administrador de la Calidad y Revisor Líder debe conocer porque permite la detección de fallas y no conformidades a través de la generación de entradas al sistema, buscando posibles fallos entre la salida esperada y la ocurrida.

1.6 Conclusiones del capítulo

El estudio de las definiciones de calidad de software permite identificar la diferencia entre la calidad del proceso de desarrollo de software y la calidad del producto software. Elemento que es primordial para esta investigación debido a que da lugar a que las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso donde juega un papel fundamental el rol de Administrador de la Calidad del CEDIN.

El análisis de las definiciones de fiabilidad permite que el autor de la investigación construya la definición que va a guiar el trabajo en los próximos capítulos.

El análisis de los ejemplos de aplicaciones de la fiabilidad de software y los resultados alcanzados en investigaciones realizadas sobre fiabilidad constituyen un factor esencial porque demuestran la relevancia del trabajo realizado y la repercusión del mismo en el DSBC (Montilva 2012).

El estudio de las técnicas de evaluación permitió identificar en qué momento del procedimiento se va a aplicar cada técnica, además de resaltar la necesidad de que el Administrador de la Calidad conozca cómo aplicar cada una de estas técnicas.

CAPÍTULO 2: PROPUESTA DE PROCEDIMIENTO PARA EVALUAR LA FIABILIDAD DE SOFTWARE

2.1 Introducción

En el presente capítulo se explica la propuesta de un procedimiento para evaluar la característica de fiabilidad en el DSBC (Montilva 2012), para ser aplicado por el grupo de gestión de la calidad del CEDIN a los proyectos desarrollados en el mismo centro. Se describen los pasos a seguir para el correcto desarrollo del procedimiento y se referencia la documentación para la realización de dicha propuesta, estos pasos contienen dentro las actividades y las tareas a realizar para culminar exitosamente y con calidad los flujos de trabajo especificados. Para el buen funcionamiento de este procedimiento, las actividades de calidad para evaluar la fiabilidad de componentes de software, deben realizarse en el orden en que se describen los pasos propuestos.

2.2 Propuesta de procedimiento para evaluar la fiabilidad

Este procedimiento está dirigido a asegurar la calidad de las producciones de software de los componentes desarrollados en el CEDIN. Vale aclarar en este caso que, actualmente en el CEDIN los componentes desarrollados se tratan como productos y han sido liberados muchos de ellos por el grupo de pruebas del CEDIN, como ejemplos de esta afirmación se conocen los componentes que se nombran a continuación: Aplicaciones_v1.0, Middleware_v1.0, Seguridad, Sonido 2D-3D, DIBI de la línea gestión industrial_v1.0 y los componentes COMM3_v1.0. En el repositorio de pruebas del CEDIN existen evidencias que demuestran la veracidad de la siguiente afirmación. Seguidamente se describe la estructura del procedimiento propuesto.

2.2.1 Nombre del procedimiento

Evaluación de la fiabilidad en el desarrollo de software basado en componentes.

2.2.2 Propósito

El propósito que se pretende con la propuesta de este procedimiento es definir los pasos a seguir para lograr alcanzar en el software desarrollado el nivel de calidad que se requiere.

2.2.3 Objetivos

Proporcionar una guía paso a paso, que permita evaluar la fiabilidad de los componentes que se desarrollan en el CEDIN.

Establecer métricas que permitan medir el grado de aceptación que posee un componente de software de acuerdo a su fiabilidad.

Ejecutar la evaluación con la presencia de un Revisor Líder y un Administrador de la Calidad.

2.2.4 Alcance

Es aplicable para asegurar la calidad de los componentes del CEDIN.

2.2.5 Referencias

Manual de procedimientos IPP-1000:2008. (IPP-1000 2008)

ISO/IEC TR 9126-2. “Métricas Externas” (ISO/IEC-TR9126-2 2003)

ISO/IEC TR 9126-3. “Métricas Internas” (ISO/IEC-TR9126-3 2003)

ISO/IEC TR 14598. “Evaluación de los productos de software” (ISO/IEC-14598 1999)

2.2.6 Responsables

Ejecuta: Administrador de la Calidad y Revisor Líder.

Responsable de su ejecución: Grupo de Aseguramiento de la Calidad (GAC) del CEDIN.

Revisa y actualiza este procedimiento: Jefe del Grupo de Gestión de la Calidad (GGC) del CEDIN.

Fiscaliza su cumplimiento: Jefe del GGC del CEDIN.

2.2.7 Términos y Definiciones

Acción Correctiva: acción tomada para eliminar la causa de una no conformidad detectada u otra actividad no deseada.

Actividad: conjunto de tareas realizadas por una persona o entidad.

Artefacto: productos tangibles de un proyecto, que pueden ser producidos y modificados por las actividades.

Auditoría: dirigida por una persona autorizada con el propósito de proporcionar una evaluación de los productos y procesos de software con el fin de evaluar la conformidad del software.

Fases: representa el ciclo de vida en el desarrollo de un software.

Iteraciones: secuencia de actividades realizadas.

Metodología: es un proceso que define quién, qué, cómo y cuándo debe realizarse una operación.

Proceso: es una definición del conjunto de actividades para transformar los requisitos que propone el cliente en un producto tangible.

Producto de software: artefacto que se crea durante el ciclo de vida de un proyecto.

Proyecto: elemento organizativo mediante el cual se gestiona el desarrollo de un software.

Riesgo: probabilidad de que ocurra un daño o una acción no deseada al sistema.

Rol: responsabilidades del equipo de trabajo de un proyecto.

Competencias: atributos personales o aptitudes que adquiere una persona para aplicar conocimientos y habilidades.

Etapa: fase de desarrollo de una acción.

Grupo de Aseguramiento de la Calidad (GAC): grupo de personas que tienen la responsabilidad de planificar la calidad, supervisarla y mantener los registros de las evidencias de las no conformidades detectadas.

Jefe de línea: encargado de asegurar la calidad en el proceso de desarrollo de software, que la aplicación producida se ajusta a las especificaciones del cliente y que está razonablemente libre de errores.

Jefe de Proyecto: encargado de la definición del proyecto, toma las decisiones, aprueba las tecnologías a utilizar, coordina y organiza las tareas que se les asignan a los miembros del proyecto, gestiona los recursos y materiales necesarios para realizar el trabajo en el proyecto y lleva a cabo todo el proceso de gestión de proyecto.

Modelo: conjunto de prácticas vinculadas a los procesos de gestión y desarrollo del proyecto.

Norma: es un documento establecido por consenso y aprobado por un organismo reconocido.

Planificación de la calidad: es necesaria para una correcta identificación de los objetivos de calidad de una organización y la forma en cómo se logran esos objetivos. Puede ayudar en el cómo medir los productos y procesos.

Revisión del producto: capacidad para soportar cambios.

Procedimiento: sucesión cronológica de operaciones concatenadas entre sí.

Responsable: obligado a responder por algo o por alguien.

RTF: revisiones técnicas formales. Es una actividad de garantía de la calidad de software.

SQA: aseguramiento de la calidad del software.

Tarea: trabajo que debe realizarse en el tiempo asignado para su desarrollo.

2.2.8 Descripción del procedimiento

Paso 1: Definir fases, iteraciones, y tareas

Para la realización de este paso es necesario conocer que el CEDIN se encuentra inmerso en el nivel 2 de CMMI, el mismo, lo que pretende es conseguir que en los proyectos de la organización haya una gestión de los requisitos y que los procesos (formas de hacer las cosas) estén planeados, ejecutados, medidos y controlados. En el capítulo 1 se explica con mejor precisión en que consiste el modelo CMMI.

A continuación se muestra una tabla donde se especifican las sub-características que puede usar cada rol en una fase determinada. Ver anexo 3 para un mayor entendimiento de la tabla 1, y ver anexo 2 para conocer las fases del ciclo de vida del proceso de mejora.

Tabla 1: Sub-características de fiabilidad en las fases de desarrollo de software según los roles.

Roles	Sub-características	Nivel requerido	Fases							
			Modelación del negocio	Requisitos	Análisis y Diseño	Implementación	Pruebas Internas	Pruebas de Liberación	Despliegue	Soporte
Revisor líder	Madurez	Alto				x	x	x	x	x
	Tolerancia ante fallos	Alto				x	x	x	x	x
	Recuperabilidad	Alto				x	x	x	x	x

Administrador de la Calidad	Madurez	Alto				x	x	x	x	x
	Tolerancia ante fallos	Alto				x	x	x	x	x
	Recuperabilidad	Alto				x	x	x	x	x

Dentro de cada fase del ciclo de desarrollo que propone CMMI en el nivel 2 es necesario tener en cuenta una serie de actividades, que se evaluarán en cada una de las fases excepto en las fases de modelación del negocio, levantamiento de requisitos y análisis y diseño, ya que aquí no es necesaria la evaluación debido a que esta fase solo realiza la descripción del componente final y se representa el análisis del negocio para el mismo.

Las actividades son:

- Consultar nombre, tipo, estado y fase del procedimiento a evaluar.
- Especificar propósito de la evaluación.
- Identificar el rol que ejecutará la evaluación.
- Identificar los atributos de fiabilidad a evaluar por fases según el rol.
- Especificar métricas.
- Establecer criterios de evaluación.
- Calcular métricas.
- Analizar atributos por niveles.
- Identificar posibles no conformidades que tenga el componente.
- Describir acciones correctivas para mitigar las no conformidades detectadas.
- Informe. Evaluación de la fiabilidad.

Paso2: Describir los roles y responsabilidades.

Revisor líder:

Responsabilidades:

- Planificar y organizar el trabajo del equipo de revisores.
- Ejecutar la revisión de evaluación de la calidad a la característica de fiabilidad del componente de software.
- Llevar a cabo la revisión dentro del horario acordado.
- Verificar la exactitud de la información recopilada.
- Confirman que la evidencia es apropiada para apoyar las no conformidades detectadas.
- Evaluar aquellos factores que puedan afectar la fiabilidad de las no conformidades y conclusiones.
- Procesar y analizar los resultados obtenidos en la revisión de evaluación de la fiabilidad.

El Revisor Líder, para un buen desempeño del rol que ocupa, debe tener conocimientos mínimos de análisis y diseño de sistemas, implementación de los conceptos relacionados con la fiabilidad de software; saber evaluar con precisión las métricas aplicadas al componente. Además de tener conocimiento de las distintas metodologías y modelos de calidad conocidos, aptitud para identificar la mejor alternativa de solución que se pueda presentar y la más fiable posible, y una de las cosas principales, una muy buena disposición para el trabajo. Se recomienda que el rol de Revisor Líder sea desempeñado por alguno de los profesores que pertenezcan al GGC debido a que este debe ser una persona con mucha experiencia desempeñando el rol de Administrador de la Calidad.

Administrador de la Calidad:

Responsabilidades:

- Elabora el plan de aseguramiento de la calidad donde incluye, cómo evaluar la fiabilidad de un componente de software.
- Elabora el plan de mediciones incluyendo métricas para evaluar la madurez, la tolerancia ante fallos y la recuperabilidad.

- Participa en la elaboración del plan de monitoreo y en el monitoreo y análisis de las áreas de proceso según la fase en la que se encuentre el componente, verificando cómo se comporta la fiabilidad del mismo.
- Guía el diseño y ejecución de las pruebas internas que se hacen a los componentes, teniendo en cuenta la fiabilidad que estos presentan.
- Participa en el análisis y recolección de los datos para aplicar las métricas de las características de fiabilidad.
- Colabora en las auditorías que se le realizan al proyecto comprobando si el componente cumple con las características de fiabilidad.
- Crea una cultura de calidad especificando la importancia que tiene la fiabilidad para los componentes que se desarrollen en el CEDIN.

El Administrador de la Calidad para realizar una exitosa evaluación de la fiabilidad debe tener ciertas habilidades para un mejor desempeño a la hora de realizar la evaluación. Debe tener amplio conocimiento de conceptos relacionados con la calidad especialmente la característica fiabilidad. Debe, principalmente, trabajar en equipo, ser buen comunicador, tener amplio poder estadístico y de síntesis. Además de dominar el ciclo de desarrollo del software y tener amplios conocimientos en la rama ingeniería de software y tomar decisiones cuando sea preciso.

Paso 3: Definir las sub-características de la fiabilidad en las fases del desarrollo de software según los roles.

La fiabilidad como característica de la calidad de software trae implícita una serie de sub-características las cuales fueron definidas en el capítulo 1.

Las sub-características son: madurez, tolerancia ante fallos, recuperabilidad.

Paso 4: Definir a partir de las normas utilizadas las métricas a seguir.

Para la confección del procedimiento se van a utilizar las normas: (ISO/IEC-9126-1 2001) e (ISO/IEC-14598 1999), para la ejecución de este paso. La primera de estas normas será empleada para especificar las métricas a utilizar según cada sub-característica de la característica de fiabilidad. Cada sub-

característica estará enfocada a ser evaluada por ambos roles (Revisor líder y Administrador de la Calidad).

A continuación se explican las métricas que se proponen aplicar para la evaluación de la fiabilidad en el DSBC.

Métricas de Madurez:

1. Nombre de la métrica: Estimado de la densidad de fallos latentes.

Esta métrica permite contar el número de defectos detectados durante las revisiones de evaluación realizadas en un periodo de tiempo y predecir el número potencial de errores futuros con un crecimiento de la fiabilidad del modelo de estimación (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X= {ABS (A1-A2)} / B

(X: estimación de la densidad residual de fallas latentes)

ABS ()= valor absoluto

A1: número total de fallos latentes que se predijo en un producto de software.

A2: número total de fallos detectados en realidad.

B: tamaño del producto.

2. Nombre de la métrica: Falta de resolución.

Esta métrica permite contar el número de fallos que no vuelvan a ocurrir durante las revisiones de evaluación en las condiciones similares (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A1/A2

A1: número de fallos resueltos.

A2: número total de fallos detectados en realidad.

3. Nombre de la métrica: Falta de densidad.

Esta métrica permite contar el número de defectos detectados y la densidad de cómputo (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A/B

A: número de fallos detectados.

B: tamaño del producto.

4. Nombre de la métrica: Eliminación de fallos.

Esta métrica permite contar el número de defectos removidos durante las revisiones y comparar ápice el número total de fallos detectados que se predijo de acuerdo al número total de averías (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A1/A2

A1: número de fallos corregidos.

A2: número total de fallos detectados en realidad.

Y=A1/A3

A3: número total de fallos latentes previsto en el producto de software.

5. Nombre de la métrica: Cobertura de la prueba (escenario de la operación se específica la cobertura de análisis).

Esta métrica permite contar el número de revisiones realizadas durante el período de evaluación y comparar el número de revisiones necesarias para obtener cobertura de la evaluación adecuada (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A/B

A: número de revisiones que realmente lleva a cabo la operación que representa el escenario durante la evaluación.

B: número de revisiones que se realiza para cubrir las necesidades.

6. Nombre de la métrica: Prueba de madurez.

Esta métrica permite contar el número de revisiones que han sido realmente ejecutados y compararlo con el número total de revisiones que se realizará según los requisitos (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A/B

A: número de revisiones que pasan durante la evaluación.

B: número de revisiones que se realiza para cubrir las necesidades.

Métricas de Tolerancia ante fallos

1. Nombre de la métrica: Evitar Desglose.

Esta métrica permite contar el número de ocurrencia de averías con respecto al número de fallos (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=1-A/B

A: número de averías.

B: número de fallos.

2. Nombre de la métrica: Evitar el fracaso.

Esta métrica permite contar el número de patrones de falla y evitar compararlo con el número de patrones de fallas para ser considerado (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A/B

A: número de ocurrencias evitar el fracaso crítico y grave contra las revisiones del patrón de culpa.

B: número de revisiones ejecutadas del patrón de culpa (casi causando insuficiencia) durante la evaluación.

3. Nombre de la métrica: Evitar un funcionamiento incorrecto.

Esta métrica permite contar el número de revisiones de funcionamientos incorrectos que evitaron causar fallos críticos y graves, y compararlo con el número de revisiones ejecutadas de los patrones de funcionamiento incorrecto a considerar (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A/B

A: número de ocurrencias evitar fallos críticos y graves.

B: número de revisiones ejecutadas de los patrones de funcionamiento incorrecto (fallos casi causa) durante la evaluación.

Métricas de Recuperabilidad

1. Nombre de la métrica: Disponibilidad.

Esta métrica permite medir el período de tiempo de reparación cada vez que el sistema no estaba disponible durante la evaluación. Calcular el tiempo medio de reparación (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

a) X= { To / (To+Tr) }

b) Y= A1/A2

To: operación de tiempo.

Tr: tiempo de reparación.

A1: total de casos de uso a disposición del usuario de software de éxito cuando el usuario intenta utilizar el sistema.

A2: total de casos de intento de usuario para utilizar el software durante la medición del tiempo.

2. Nombre de la métrica: Significa el tiempo de inactividad.

Esta métrica permite medir el tiempo en el cual el sistema no está disponible durante un período de evaluación especificado y calcula el tiempo medio (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=T/N

T: tiempo de inactividad total.

N: número de averías observadas.

3. Nombre de la métrica: El tiempo medio de recuperación.

Esta métrica permite medir los tiempos de recuperación completa. Para cada parte del tiempo se ha presentado el sistema hacia abajo durante el período de evaluación especificado y calcular el tiempo medio (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de elementos de los datos:

X= Sum (T)/B

T: tiempo para la recuperación de caídas del sistema de software en cada oportunidad.
B: número de casos que observó del software del sistema entró en la recuperación.

4. Nombre de la métrica: Restaurar la eficacia.

Esta métrica permite contar la cantidad de tiempo de restauración de evaluación y se compara con el número de restauraciones necesarias con el tiempo de destino especificado (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A/B

A: número de casos restaurado con éxito para alcanzar la meta de tiempo de restauración.
B: número de casos plasmados.

5. Nombre de la métrica: Restaurabilidad.

Esta métrica permite contar el número de restauraciones exitosas y compararlo con el número de la restauración de evaluaciones requerida (ISO/IEC-TR9126-2 2003).

Fórmula de medición y cálculo de los elementos de los datos:

X=A/B

A: número de casos de restauración hecho con éxito.

B: número de casos probados de restauración según los requisitos.

Obtener valor de una sub-característica específica

El objetivo de esta métrica es conocer el valor de la sub-característica, debido a que hay varios casos donde existe más de una métrica para la misma sub-característica. Se pretende con esto conocer un único valor.

$$Vsub = \frac{\sum \text{CmSub}}{CM}$$

Vsub: valor de la sub-característica.

CmSub: valor obtenido después de haber realizado el cálculo de la métrica para esa sub-característica.

CM: cantidad de métricas de la sub-característica.

Por ciento de fiabilidad

Con esta métrica se calcula el por ciento de fiabilidad de un componente con respecto a las sub-características escogidas esta fórmula depende del resultado anterior calculado de la fórmula.

$$Pfi = \frac{\sum \text{VSub}}{\text{CSub}} * 100$$

Pfi: por ciento fiabilidades.

VSub: valor de la sub-característica.

CSub: cantidad de sub-características.

Nivel de fiabilidad

Una vez aplicada las fórmulas anteriores se necesitan conocer qué nivel y por ciento finales de fiabilidad posee el componente, para ello se utiliza la tabla 2 que se presenta a continuación:

Tabla 2: Niveles de Fiabilidad.

Nivel	Intervalo	Porciento Final
Alto	**0.8 ≤ X ≤ 1**	**80% ----100%**
Medio	**0.5 ≤ X < 0.8**	**50%----79%**
Bajo	**X < 0.5**	**Menor o igual que 49%**

Paso 5: Valorar resultados

En este paso se definen una serie de indicadores que se dan como solución a las métricas propuestas en el paso anterior.

Tabla 3: Valoración de los resultados.

Indicadores de Métricas		
Madurez		
No	**Nombre de la métrica**	**Interpretación del valor medido**
1	Estimado de la densidad de fallos latentes	0<=X Depende de las revisiones realizadas durante la evaluación. En las últimas etapas, mientras más pequeño es mejor.
2	Falta de resolución	0 <= X<=1, el más cercano a 1.0 es mejor a medida que más fracasos se resuelven.
3	Falta de densidad	0<=X Depende de las revisiones realizadas durante la evaluación. En las últimas etapas, mientras más pequeño es mejor.
4	Eliminación de fallos	0 <= X<=1, el más cercano a 1.0 es mejor mientras menos fallos continúen existiendo. 0<=Y, el más cercano a 1.0 es mejor mientras menos fallos continúen existiendo.
5	Cobertura de la prueba (escenario de la operación se específica la cobertura de análisis)	0 <= X<=1, el más cercano a 1.0 es la mejor cobertura de la prueba.
6	Prueba de madurez	0 <= X<=1, el más cercano a 1.0 es la mejor.
Tolerancia ante fallos		
1	Evitar Desglose	0 <= X<=1, el más cercano a 1.0 es la mejor.

2	Evitar el fracaso	0 <= X<=1, el más cercano a 1.0 es mejor debido a que el usuario puede evitar con más frecuencia que se produzcan fallos críticos o graves.
3	Evitar un funcionamiento incorrecto	0 <= X<=1, el más cercano a 1.0 es mejor debido a que se evita una operación más de usuario incorrecto.
	Recuperabilidad	
1	Disponibilidad	0 <= X<=1, el más grande y más cercano a 1.0 es mejor que el usuario puede utilizar el software para obtener más tiempo. 0 <= Y<=1, el más grande y más cercano a 1.0 es la mejor.
2	Significa el tiempo de inactividad	0<X, más pequeño es él es el mejor, el sistema estará fuera de servicio por un tiempo más corto.
3	El tiempo medio de recuperación	0<X, cuanto más pequeño es el mejor.
4	Restaurar la eficacia	0 <= X<=1, el más grande y más cercano a 1.0 es la mejor, ya que el proceso de restauración en el producto es más eficaz.
5	Restaurabilidad	0 <= X<=1, el más grande y más cercano a 1.0 es mejor ya que producto es más capaz de restaurar en los casos definidos.

Paso 6: Incluir el procedimiento al plan de trabajo del GGC.

En este paso lo que se pretende es añadir en el plan de trabajo del GAC el procedimiento propuesto para medir la fiabilidad de un componente de software en el CEDIN.

Paso 7: Describir las actividades a desarrollar según el proceso de mejora.

1. Consultar nombre, tipo, estado y fase del procedimiento a evaluar.

Se consulta el nombre, tipo, estado y fase del procedimiento a evaluar a través de una entrevista realizada al jefe de proyecto donde se encuentren los componentes a evaluar.

El estado del componente puede ser de dos formas: en desarrollo o terminado.

Se selecciona la fase en la que se encuentra el componente. Las fases que se van a utilizar son las que propone el ciclo de vida del programa de mejora, debido a que en el CEDIN la mayoría de los proyectos se encuentran inmersos en el programa de mejora y el mismo se encuentra en el nivel 2 de CMMI (CMMI-SE/SW/IPPD/SS. 2002). Es muy probable que existan proyectos en los que sus componentes se hayan desarrollado o se están desarrollando con una metodología, en estos casos específicos hay que especificar la fase en la que se encuentra y la metodología que usa. Si el proyecto donde se está desarrollando el componente está inmerso en el programa de mejora, se especifica en la fase que se

encuentra según las que estén definidas en el mismo. Para ver más detalladamente el ciclo de vida definido en el Proceso de Mejora Ver Anexo 2.

2. Especificar propósito de la evaluación.

El propósito de la evaluación de la característica de fiabilidad es detectar las no conformidades o fallos que puedan existir evaluando las sub-características (madurez, tolerancia ante fallos, recuperabilidad) de la misma, definiéndolas y mitigando dichos fallos mediante acciones correctivas para el mejoramiento de dicho componente.

3. Identificar el rol que ejecutará la evaluación.

Se identifica el rol encargado de realizar la evaluación de la fiabilidad, solo hay dos roles que tienen estos permisos el Revisor Líder y el Administrador de la Calidad.

Las responsabilidades del Revisor Líder y el Administrador de la Calidad se pueden ver en el Capítulo 2, epígrafe 2.2.8, Paso 2, las mismas fueron referenciadas del documento 0516_Roles y Responsabilidades que se encuentra publicado en http://calisoft.uci.cu, y adaptadas durante esta investigación a las necesidades que deben tener estos dos roles para una correcta evaluación de la fiabilidad de un componente de software.

4. Identificar los atributos de fiabilidad a evaluar por fases según el rol.

El rol encargado de la evaluación se encargará de identificar los atributos a evaluar en la fase en que se encuentren, los atributos de la fiabilidad se encuentran definidos en el capítulo 1. Es de gran importancia que se realice una correcta identificación y de ser posible evaluar todas las sub-características ya que todas tienen un alto grado de importancia para la evaluación de cualquier software.

5. Especificar métricas.

Se especifican las métricas de los atributos de la fiabilidad necesarias para realizar una correcta evaluación, las mismas se encuentran definidas en el capítulo 2, epígrafe 2.2.8, paso 4. Definir a partir de las normas utilizadas las métricas a seguir.

6. Establecer criterios de evaluación.

Se especifican los criterios para poder evaluar el resultado de las métricas aplicadas al procedimiento y luego llegar a una conclusión, para ello ver capítulo 2, epígrafe 2.2.8, paso 5. Valorar resultados.

7. Analizar atributos por niveles.

Se procede a otorgarle niveles a cada sub-característica después de haber evaluado las métricas asociadas a ellas, estos niveles se seleccionan según los definidos en la tabla 8.

La tabla que se muestra a continuación resume todo el proceso de la medición. Ver especificaciones de la tabla en anexo 4.

Tabla 4: Ejecución de la evaluación.

Sub-característica	Nivel Requerido	Resultados	
		Valor Métrica	Nivel Real
Porciento de fiabilidad			

8. Identificar posibles no conformidades que tenga el componente.

Se identifican los posibles errores que pueda tener el procedimiento a evaluar para ello ver que se encuentran definidos en el capítulo 2, epígrafe 2.2.8, paso 8. Identificar posibles errores a detectar.

9. Describir acciones correctivas para mitigar las no conformidades detectadas.

Se describen las acciones correctivas para mitigar los errores más comunes que puedan existir para ello ver capítulo 2, epígrafe 2.2.8, paso 9. Describir acciones correctivas que se propondrían para dar solución a los errores planteados.

10. Informe evaluación de la fiabilidad.

Este es el artefacto por donde se le da salida al procedimiento, donde se documenta todo el proceso de la evaluación, en el anexo 5 se muestra como queda la estructura.

Paso 8: Identificar posibles errores a detectar.

- Que la métrica no se pueda aplicar.
- Que la sub-características de la característica no se pueda evaluar.
- Que no se encuentre el documento “Procedimiento de la fiabilidad” para la evaluación del componente.
- Que el Revisor Líder o el Administrador de la Calidad no estén capacitados o no tengan conocimientos de la fiabilidad del software.
- Que el evaluador no conozca cómo funciona el componente que va a evaluar.

Paso 9: Describir acciones correctivas que se propondrían para dar solución a los errores planteados.

- Proponer métricas sencillas y visibles para que no exista ningún tipo de complicación a la hora de realizarlas.
- Comprobar que cada sub-característica tiene una medida a tener en cuenta para la evaluación.
- Publicar el documento como unos de los documentos de apoyo que aparecen en la página del CEDIN.
- Capacitar a los administradores de calidad y a los revisores líderes para que conozcan cómo funciona la característica de fiabilidad y en específico el procedimiento a seguir para evaluar dicha característica.
- El evaluador antes de aplicar el procedimiento a seguir, debe capacitarse en cómo funciona el producto que va a evaluar para lograr que el producto se libere con la calidad requerida.

2.3 Conclusiones del Capítulo

La propuesta de procedimiento para evaluar la fiabilidad de un componente de software permite establecer el nivel de fiabilidad de un componente, a partir de ciertos parámetros que fueron planteados en el transcurso del capítulo, siendo éste el punto de partida para realizar las revisiones de evaluación a los componentes de software del CEDIN antes de que estos pasen a la fase de pruebas internas.

CAPÍTULO 3: VALIDACIÓN DEL PROCEDIMIENTO

3.1 Introducción

En este capítulo se realiza la validación del procedimiento propuesto en el capítulo anterior. Se efectúa en dos componentes de software utilizados en proyectos de informática industrial y en un componente utilizado en proyectos de realidad virtual del CEDIN. Esta validación se realiza en las versiones de estos componentes antes de que fueran integrados a los sistemas de los que iban a formar parte para ser liberados por el grupo de pruebas del CEDIN, en el 2011. Los dos primeros componentes evaluados pertenecen al Sistema de Manejo Integral para Perforación de Pozos Petroleros (SIPP) ellos son: componente “Archivo” y “Gráficas”. El tercer componente es utilizado en todos los proyectos de la línea de productos de realidad virtual, es conocido como componente de “Sonido 2D-3D”. La validación se aplica a cada componente por separado y luego se realiza una comparación de los tres.

3.2 Descripción del procedimiento en el proyecto SIPP

El procedimiento es aplicable a dos componentes de software que se utilizan en el proyecto SIPP, ambos componentes son reutilizables.

El propósito de la evaluación es detectar las no conformidades (NC) de tipo crítico que puedan tener las sub-características (madurez, tolerancia ante fallos y recuperabilidad) de fiabilidad de los componentes de software a evaluar y proponer acciones correctivas para darle solución a las NC propuestas. Además de conocer el porciento de fiabilidad que posee el componente.

El rol encargado de ejecutar el procedimiento para los dos componentes propuestos es el Administrador de la Calidad y los atributos a evaluar son: madurez, tolerancia ante fallos, recuperabilidad.

Según las sub-características y el rol definido anteriormente se especifican las métricas necesarias de cada una de las sub-características especificadas en la actividad anterior para evaluar la fiabilidad de los componentes de software y los indicadores definidos por cada una de ellas.

Tabla 5: Métricas para evaluación.

Sub-características	Nombre de las métricas escogidas para realizar la evaluación	Indicadores por métrica
Madurez	Prueba de madurez	0 <= X<=1, el más cercano a 1.0 es el mejor.
Tolerancia ante fallos	Evitar desglose	0 <= X<=1, el más cercano a 1.0 es el mejor.
	Evitar funcionamiento incorrecto	0 <= X<=1, el más cercano a 1.0 es mejor debido a que se evita una operación más de usuario incorrecto.
Recuperabilidad	Disponibilidad	0 <= X<=1, el más grande y más cercano a 1.0 es mejor debido a que el usuario puede utilizar el software para obtener más tiempo. 0 <= Y<=1, el más grande y más cercano a 1.0 es el mejor.
	Restaurar la eficacia	0 <= X<=1, el más grande y más cercano a 1.0 es el mejor, ya que el proceso de restauración en el producto es más eficaz.
_____	Porciento de fiabilidad	

La información descrita anteriormente, está definida en el capítulo 2 como actividades del procedimiento para cada uno de los componentes, se decidió redactarlo general debido a que coincide la información en los dos casos, posteriormente se trabaja cada componente por separado.

3.2.1 Componente Archivo

Consultar nombre, tipo, estado y fase del procedimiento a evaluar

Después de haber realizado la entrevista al líder de proyecto se arrojaron los siguientes resultados:

Nombre del componente de software: “Archivo”.

Tipo de componente de software: componentes imperativos (módulos) que para su desarrollo utilizan el lenguaje de programación orientado a objetos.

Estado del componente de software: terminado.

El proyecto productivo donde se desarrolló el componente se encuentra inmerso en el programa de mejora por lo tanto se rigen por el ciclo de vida que define el mismo, CMMI nivel 2 y actualmente se encuentra en la fase de pruebas de liberación. Sin embargo, la validación del procedimiento propuesto se realizó en la versión del componente antes de que este pasara a la fase de pruebas internas. Estas son las pruebas que realiza el grupo de pruebas del CEDIN.

Archivo es un componente que le permite al usuario de forma clara exportar todos los datos que se encuentren almacenados en la base de datos, en archivos *.xls* o en *.txt*, se selecciona el reporte que se quiere exportar en tiempo real y el día en que se quiere realizar el reporte.

Calcular métricas

Madurez

Nombre de la métrica: Prueba de madurez.

X=A/B =16/18=0.89

Se calculó la cantidad de revisiones que realmente se llevan a cabo con respecto al número de pruebas que el usuario debe realizar para cubrir las necesidades del componente respecto a su fiabilidad. Para ello se realizó evaluación al componente “Archivo”, donde cada una de las secciones a evaluar son los requisitos del componente, en este caso se plantea un requisito general que sería “Gestionar trabajo con archivo” este requisito está divido en dos requisitos específicos los cuales son: exportar archivo e importar archivo.

Se probaron 8 funcionalidades del componente permitiendo contar la cantidad de evaluaciones que fueron realmente ejecutadas, comparándolo con el número total de evaluaciones que deberían realizarse según los requisitos. Esto arrojó que el componente posee un alto nivel de madurez, el mismo es capaz de evitar un fallo total como resultado de haberse producido un fallo de software en la interfaz.

Tolerancia ante fallos

Nombre de la métrica: Evitar desglose.

X=1-A/B =1- 1/3 = 1- 0.33 = 0.67

Al realizar las evaluaciones a este componente ocurrió la siguiente avería:

- Se fue la corriente en los laboratorios y la planta estaba fragmentada.

Al realizar las evaluaciones en las 8 funcionalidades correspondientes después de la avería acontecida se detectan los siguientes fallos:

- Luego de llenar todos los datos y seleccionar exportar, el sistema muestra un mensaje indicando. Seleccione algún elemento del campo "Partes".
- Luego de llenar todos los datos y seleccionar exportar, el sistema muestra un mensaje indicando. Seleccione algún elemento del campo "Pozo".
- Luego de llenar todos los datos y seleccionar exportar, el sistema no muestra nada solamente una página con este error: *500 | Internal Server Error | sfRenderException The template "exportarSuccess.php" does not exist or is unreadable in ""*

Nombre de la métrica: Evitar funcionamiento incorrecto.

X=A/B = 3/3= 1

Para la evaluación de esta métrica, se calcula el número de ocurrencias para evitar fallos críticos con respecto al número de revisiones ejecutadas de los patrones de funcionamiento incorrecto, durante la evaluación, entiéndase como patrones de funcionamiento incorrecto los requisitos que puedan ocasionar fallos que conlleven a desastres en el sistema.

Para ello se probaron las mismas 8 funcionalidades que se utilizaron para probar la madurez constatando que cuando se evaluó ocurrieron tres fallos, en las tres revisiones ejecutadas de los patrones de funcionamiento incorrecto por lo cual la métrica dio un valor alto.

Obtener valor de una sub-característica específica

$Vsub = \frac{\Sigma \text{ CmSub}}{CM}$ = (0.67+1)/2=0.83

El componente posee un nivel alto de tolerancia ante fallos, a la hora de mantener un nivel de ejecución adecuado en caso de ocurrir un fallo de software o una infracción en la interfaz.

Recuperabilidad

Nombre de la métrica: Disponibilidad.

X= { To / (To+Tr) } = 1440/(1440+60) = 1440/1500 = 0.96

Y= A1/A2 =1/1 =1

Para la evaluación de esta métrica, fue necesario calcular el periodo de tiempo que el desarrollador tardaría en corregir los errores encontrados en el componente, para poder calcular el total de requisitos a disposición del usuario que tuvieron éxito cuando el mismo interactuó con el sistema con respecto al total de intentos del usuario de interactuar con el sistema en ese periodo de tiempo en que se corrigieron los fallos detectados.

Se probaron las mismas 8 funcionalidades que se utilizaron con las métricas anteriores logrando comprobar que el software tiene un alto nivel de disponibilidad ya que se midió el período de tiempo de reparación cada vez que el sistema no estaba disponible.

Nombre de la métrica: Restaurar la eficacia.

X=A/B = 3/3 = 1

Para la evaluación de esta métrica se midieron los defectos restaurados con éxito en el tiempo de restauración, con respecto al número de casos realizados, dicho tiempo fue dado en un periodo de 24 horas, en cada caso se resolvió el problema en menos tiempo del acordado para la restauración del sistema. Comprobando que el componente posee un alto nivel en cuanto se habla e restaurar la eficacia del mismo.

Obtener valor de una sub-característica específica.

$Vsub = \frac{\Sigma \text{ CmSub}}{CM}$ = (1+1)/2= 2/2=1

El componente posee un alto nivel de recuperabilidad, es decir recupera y restablece los datos en cuanto ocurre un fallo en el software.

Porciento de fiabilidad del componente Archivo

$Pfi = \frac{\Sigma \text{ VSub}}{\text{CSub}} * 100$ = (0.89+0.83+1)/3*100= 2.725/3=0.908*100=90.8%

Tabla 6: Ejecución de evaluación en el componente Archivo.

Sub-característica	Nivel Requerido	Resultados	
		Valor Métrica	Nivel Real
Madurez	Alto	0.89	Alto
Tolerancia ante fallos	Alto	0.83	Alto
Recuperabilidad	Alto	1	Alto
Porciento de fiabilidad		90.8%	

Analizar posibles Resultados

En la tabla se muestran los resultados de la evaluación que se le realizó al componente “Archivo” perteneciente al proyecto SIPP. Se muestra el nivel que requiere cada sub-característica y el nivel real que poseen, no denotando resultados negativos en ningún caso al realizar las revisiones al componente. Por lo tanto se concluye que dicho componente es maduro, tolerante ante fallos y recuperable con un por ciento alto de fiabilidad.

No conformidades detectadas

El componente posee un alto por ciento de fiabilidad sin embargo se encontraron una serie de fallos los cuales fueron resueltos en el momento.

Fallos encontrados después de haber existido una avería en el sistema:

Tabla 7: No conformidades y acciones correctivas encontradas en el componente Archivo.

Fallos encontrados	Acciones correctivas	Solucionados
Luego de llenar todos los datos y seleccionar exportar, el sistema muestra un mensaje indicando. Seleccione algún elemento del campo "Partes".	Revisar el código del componente donde da la opción de exportar, y revisar la validación de manera que se le dé solución al problema planteado.	Resuelta
Luego de llenar todos los datos y seleccionar exportar, el sistema muestra un mensaje indicando. Seleccione algún elemento del campo "Pozo".	Revisar el código del componente donde da la opción de exportar, y revisar la validación de manera que se le dé solución al problema planteado.	Resuelta
Luego de llenar todos los datos y seleccionar exportar, el sistema no muestra nada solamente una página con este error ***500 | Internal Server Error | sfRenderException The template "exportarSuccess.php" does not exist or is unreadable in "***	Revisar el código del componente al parecer se des-configuró la implementación realizada.	Resuelta

3.2.2 Componente Gráficas

Consultar nombre, tipo, estado y fase del procedimiento a evaluar

Después de haber realizado la entrevista al líder de proyecto se arrojaron los siguientes resultados:

Nombre del componente de software: “Gráficas”.

Tipo de componente de software: componentes imperativos (módulos) que para su desarrollo utilizan el lenguaje de programación orientado a objetos.

Estado del componente de software: terminado.

El componente se encuentra en la misma fase del componente anterior debido a que ambos se trabajan en el proyecto SIPP.

Calcular métricas

Madurez

Nombre de la métrica: Prueba de madurez.

X=A/B = 2/3 = 0.67

Se calculó la cantidad de revisiones que realmente se llevan a cabo con respecto al número de pruebas que el usuario debe realizar para cubrir las necesidades del componente respecto a su fiabilidad. Para ello se realizó una evaluación al componente “Gráficas”, donde cada una de las secciones a evaluar son los requisitos del componente, en este caso se plantea un requisito general que sería “Generar gráficas de indicadores” este requisito está divido en requisitos específicos los cuales son: graficar metraje diario del pozo; graficar perforación diaria del pozo; graficar perforación vertical del pozo; graficar perforación horizontal del pozo; graficar comportamiento de los gastos diarios del pozo; graficar resumen del tiempo por etapas de perforación.

Se probaron 7 funcionalidades del componente permitiendo contar la cantidad de evaluaciones que fueron realmente ejecutadas, comparándolo con el número total de revisiones que deberían realizarse según los requisitos. Esto arrojó como resultado que el componente posee un nivel medio de madurez.

Tolerancia ante fallos

Nombre de la métrica: Evitar desglose.

X=1-A/B = 1- 1/1 =1-1=0

Al realizar las evaluaciones a este componente ocurrió la siguiente avería:

- Se reinició la computadora sin tener en cuenta que esta funcionaba como servidor del proyecto SIPP.

Al probarse las 7 funcionalidades después de haber ocurrido la avería se detectó el siguiente fallo:

- Cuando se selecciona algún tipo de gráfica da un error de php, en vez de mostrar la interfaz correspondiente. El error que se muestra es Error: *An invalid or illegal string was specified.*

En el caso de esta métrica da valores bajos debido a que se encontró un fallo en el componente cuando levanto el sistema que puede causar que no se muestren las gráficas que se deseen mostrar.

Nombre de la métrica: Evitar funcionamiento incorrecto.

X=A/B = 1/1= 1

Para la evaluación de esta métrica se requiere calcular el número de ocurrencias para evitar fallos críticos con respecto al número de revisiones ejecutadas de los patrones de funcionamiento incorrecto durante la evaluación.

Se probaron 7 funcionalidades donde se detectó la existencia de un fallo crítico en el componente con respecto a la evaluación ejecutada de los patrones de funcionamiento incorrecto obteniendo resultados destacados en la métrica.

Obtener valor de una sub-característica específica

$Vsub = \frac{\sum CmSub}{CM}$ = (0+1)/2= 1/2= 0.5

El componente posee un nivel medio de tolerancia ante fallos, a la hora de mantener un nivel de ejecución adecuado en caso de ocurrir un fallo de software o una infracción en la interfaz.

Recuperabilidad

Nombre de la métrica: Disponibilidad.

X= { To / (To+Tr) } = 1440/(1440+60) = 1440/1500 = 0.96

Y= A1/A2 =1/1 =1

Para la evaluación de esta métrica fue necesario calcular el período de tiempo que el desarrollador tardaría en corregir los fallos encontrados en el componente para poder calcular el total de requisitos a

disposición del usuario que tuvieron éxito cuando el mismo interactuó con el sistema con respecto al total de intentos del usuario de interactuar con el sistema en ese período de tiempo en que se corrigieron los fallos detectados.

Se probaron 7 funcionalidades logrando comprobar que el software tiene un alto nivel de disponibilidad ya que se midió el período de tiempo de reparación cada vez que el sistema no estaba disponible.

Nombre de la métrica: Restaurar la eficacia.

X=A/B = 2/2 = 1

Para la evaluación de esta métrica se midió los defectos restaurados con éxito en el tiempo de restauración, con respecto al número de casos realizados, dicho tiempo fue dado en un periodo de 24 horas, en cada caso se resolvió el problema en menos tiempo del acordado para la restauración del sistema, comprobando que el componente posee un alto nivel en cuanto se habla e restaurar la eficacia del mismo.

Obtener valor de una sub-característica específica.

$Vsub = \frac{\Sigma \text{CmSub}}{CM}$ =(1+1)/2= 2/2=1

El componente posee un alto nivel de recuperabilidad, es decir, recupera y restablece los datos en cuanto ocurre un fallo en el software.

Porciento de fiabilidad del componente Gráficas

$Pfi = \frac{\Sigma \text{VSub}}{\text{CSub}} * 100$ =(0.67+0.5+1)/3*100=2.17/3*100=0.72*100=72%

Tabla 8: Ejecución de evaluación en el componente Gráficas.

Sub-característica	Nivel Requerido	Resultados	
		Valor Métrica	Nivel Real
Madurez	Alto	0.67	Medio
Tolerancia ante fallos	Alto	0.5	Medio
Recuperabilidad	Alto	1	Alto
Porciento de fiabilidad		72%	

Analizar posibles Resultados

En la tabla se muestran los resultados de la evaluación que se le hizo al componente “Gráficas” perteneciente al proyecto SIPP, se muestra el nivel que requiere cada sub-característica y el nivel real que poseen, no denotando resultados negativos en ningún caso, a pesar de encontrarse fallos en el software, por lo que se concluye que dicho componente es recuperable. Sin embargo, debido al nivel medio alcanzado en el caso de la madurez y la tolerancia ante fallos, el mismo alcanza como resultado que posee un por ciento medio de fiabilidad.

Fallos encontrados en el sistema:

A pesar de que el componente posee un por ciento medio de fiabilidad se encontró un fallo el cual fue resuelto en el período de tiempo estimado.

Fallo encontrado después de haber existido una avería en el sistema:

Tabla 9: No conformidades y acciones correctivas encontradas en el componente Gráficas.

Fallos encontrados	Acciones correctivas	Solucionados
Cuando se selecciona algún tipo de gráfica da un error de php, en vez de mostrar la interfaz correspondiente. El error que se muestra es: Error: *An invalid or illegal string was specified.*	Revisar el código del componente donde se deben mostrar las gráficas, y colocar la opción correcta para que se muestre el gráfico sin problemas.	Resuelto

3.3 Descripción del procedimiento en el Componente Sonido 2D-3D

El procedimiento es aplicable al componente de software Sonido 2D-3D, el mismo es reutilizable.

El propósito de la evaluación es detectar las no conformidades (NC) de tipo crítico que puedan tener las sub-características (madurez, tolerancia ante fallos y recuperabilidad) de fiabilidad de los componentes de software a evaluar y proponer acciones correctivas para darle solución a las NC propuestas. Además de conocer el porciento de fiabilidad que posee el componente.

El rol encargado de ejecutar el procedimiento para los dos componentes propuestos es el Administrador de la Calidad y los atributos a evaluar son: madurez, tolerancia ante fallos y recuperabilidad.

Según las sub-características y el rol definido, anteriormente, se especifican las mismas métricas evaluadas en los dos componentes anteriores.

Consultar nombre, tipo, estado y fase del procedimiento a evaluar

Después de haber realizado la entrevista al desarrollador del componente en la línea de laboratorios virtuales se arrojaron los siguientes resultados:

Nombre del componente de software: “Sonido 2D-3D”.

Tipo de componente de software: componentes imperativos (módulos) que para su desarrollo utilizan el lenguaje de programación orientado a objetos.

Estado del componente de software: terminado.

La línea de producto donde se desarrolló el componente se encuentra inmersa en el programa de mejora, por lo tanto se rigen por el ciclo de vida que define el mismo, CMMI nivel 2 y se encuentra en la fase de Despliegue.

Sonido 2D-3D es un componente que su función es integrarse a otros componentes permitiendo que se vinculen cada una de las funcionalidades del mismo al sistema en que se va a trabajar.

Calcular métricas

Madurez

Nombre de la métrica: Prueba de madurez.

X=A/B =7/14=0.5

Se calculó el número de evaluaciones que realmente se llevan a cabo con respecto al número de pruebas que el usuario debe realizar para cubrir las necesidades del componente respecto a su fiabilidad. Para ello se realizó una revisión al componente “Sonido 2D-3D”, donde cada una de las secciones a evaluar son los requisitos del componente.

Se probaron 12 funcionalidades del componente permitiendo contar la cantidad de revisiones que fueron realmente ejecutadas, comparándolo con el número total de revisiones que deberían realizarse según los requisitos. Esto arrojó como resultado que el componente posee un nivel medio de madurez, el mismo es capaz de evitar un fallo total como resultado de haberse producido un fallo de software en la interfaz.

Tolerancia ante fallos

Nombre de la métrica: Evitar desglose.

X=1-A/B =1- 0/0 = 1-0 = 1

Al realizar la evaluación en las 12 funcionalidades del componente no ocurrieron averías al sistema, y no se detectaron fallos en ninguna de ellas por lo que la evaluación realizada dio resultados satisfactorios.

Nombre de la métrica: Evitar funcionamiento incorrecto.

X=A/B

Para la evaluación de esta métrica se requiere calcular el número de ocurrencias para evitar fallos críticos con respecto al número de revisiones ejecutadas de los patrones de funcionamiento incorrecto durante la evaluación, entiéndase como patrones de funcionamiento incorrecto los requisitos que puedan ocasionar fallos que conlleven a desastres en el sistema.

Para ello se probaron 12 funcionalidades constatando que cuando se evaluó no ocurrieron fallos, por lo mismo esta métrica no puede ser evaluada.

El componente posee un nivel medio de tolerancia ante fallos, a la hora de mantener un nivel de ejecución adecuado en caso de ocurrir un fallo de software o una infracción en la interfaz, debido a que no se evaluó la métrica de ¨Evitar funcionamiento incorrecto¨ se toman solamente los valores de la métrica anterior.

Recuperabilidad

Nombre de la métrica: Disponibilidad.

X= { To / (To+Tr) } = 1440/(1440+60) = 1440/1500 = 0.96

Y= A1/A2 =1/1 =1

Para la evaluación de esta métrica se calculó el período de tiempo que el desarrollador tardaría en corregir los errores encontrados en el componente para poder calcular el total de requisitos a disposición del usuario que tuvieron éxito cuando el mismo interactuó con el sistema con respecto al total de intentos del usuario de interactuar con el sistema en ese período de tiempo en que se corrigieron los fallos detectados.

Se probaron las mismas 12 funcionalidades que se utilizaron con las métricas anteriores logrando comprobar que el software tiene un alto nivel de disponibilidad ya que se midió el período de tiempo de reparación cada vez que el sistema no estaba disponible.

Nombre de la métrica: Restaurar la eficacia.

X=A/B

Para la evaluación de esta métrica se midió los fallos restaurados con éxito en el tiempo de restauración, con respecto al número de casos realizados, dicho tiempo fue dado en un periodo de 24 horas. En cada caso se resolvió el problema en menos tiempo del acordado para la restauración del sistema.

Esta métrica no puede ser evaluada debido que no se encontraron fallos en el componente, por lo que se toma solamente el valor de la métrica anterior.

El componente posee un alto nivel de recuperabilidad, es decir recupera y restablece los datos en cuanto ocurre un fallo en el software.

Porciento de fiabilidad del componente Sonido 2D-3D

$Pfi = \frac{\Sigma\, VSub}{CSub} * 100$ = (0.5+1+1)/3*100=2.5/3*100= 0.83*100=83%

Tabla 10: Ejecución de evaluación en el componente Sonido 2D-3D.

Sub-característica	Nivel Requerido	Resultados	
		Valor Métrica	Nivel Real
Madurez	Alto	0.5	Medio
Tolerancia ante fallos	Alto	1	Alto
Recuperabilidad	Alto	1	Alto
Porciento de fiabilidad		83%	

Analizar posibles Resultados

En la tabla se muestran los resultados de la evaluación que se le hizo al componente “Sonido 2D-3D” perteneciente a la línea de producto de Realidad Virtual, se muestra el nivel que requiere cada sub-característica y el nivel real que poseen, no denotando resultados negativos en ningún caso, a pesar de que la madurez alcanzó un nivel medio al realizar las evaluaciones al software. Por lo que se concluye que dicho componente es maduro, tolerante ante fallos y recuperable dando como resultado que posee un alto por ciento de fiabilidad.

3.4 Comparación de Resultados

El componente “Sonido 2D-3D” se encuentra en estado terminado y está integrado y listo para usarse con éxito. Este componente es maduro, tolerante a fallos y recuperable por lo que se llegó a la conclusión de que posee un alto por ciento de fiabilidad. Ver fig. 3 y fig. 4

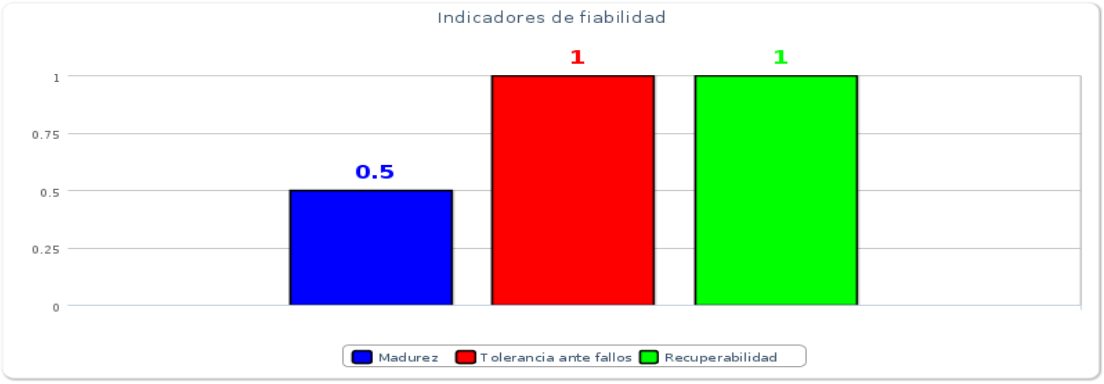

Fig. 3: Indicadores de fiabilidad del componente Sonido 2D-3D.

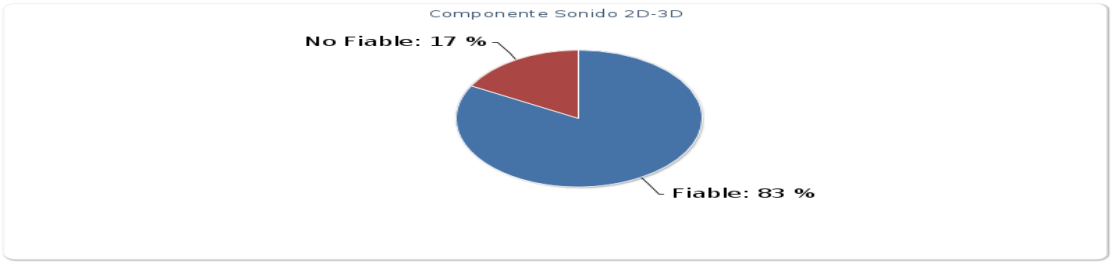

Fig. 4: Por ciento de fiabilidad del componente Sonido 2D-3D.

En los componentes “Archivo” y “Gráficas” se encontraron fallos significativos que pueden atentar contra la fiabilidad del sistema en general, a pesar de que los fallos fueron resueltos en el tiempo requerido. El componente “Archivo” alcanzó un por ciento alto de fiabilidad por lo que este componte es maduro, tolerante a fallos y recuperable. Ver fig. 5 y fig. 6.

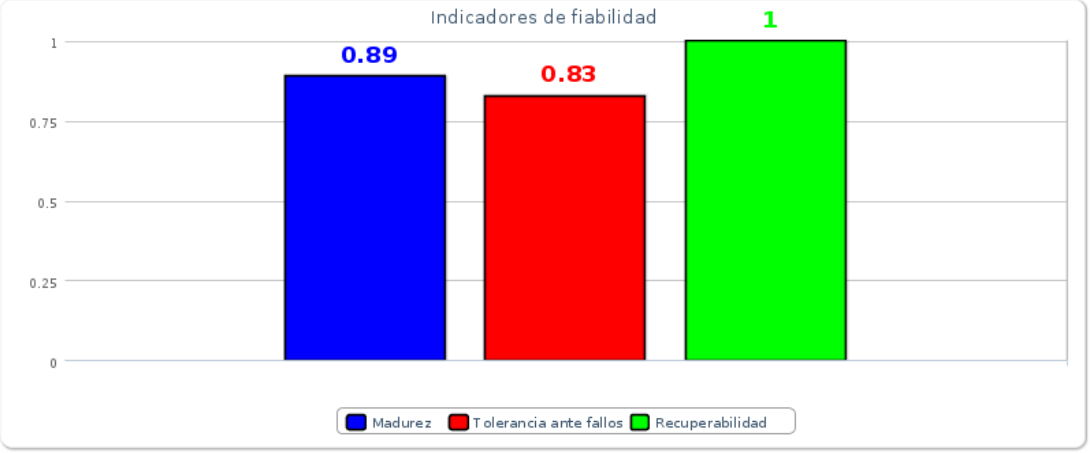

Fig. 5: Indicadores de fiabilidad del componente Archivo.

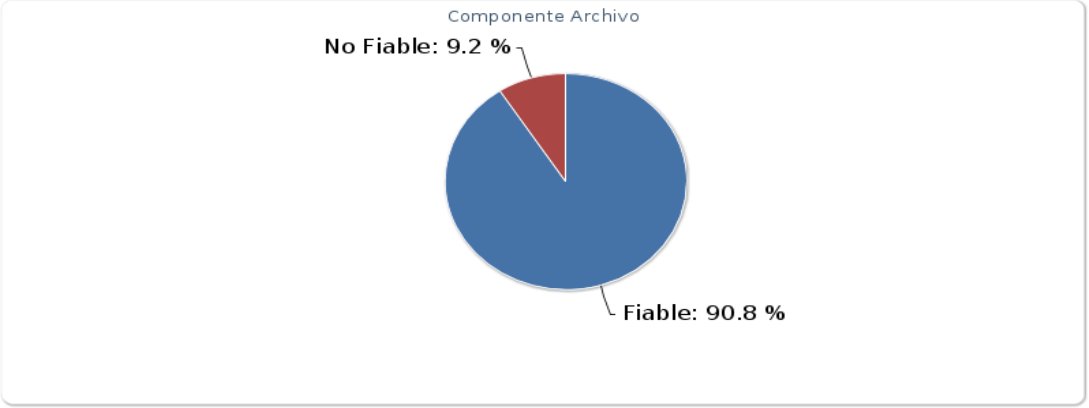

Fig. 6: Por ciento de fiabilidad del componente Archivo.

Sin embargo el componente “Gráficas”, a pesar de pertenecer al mismo proyecto se pudo constatar que es recuperable, pero presenta problemas con las demás sub-características por lo que se llega a la conclusión que poseen un por ciento medio de fiabilidad. Ver fig. 7 y fig. 8.

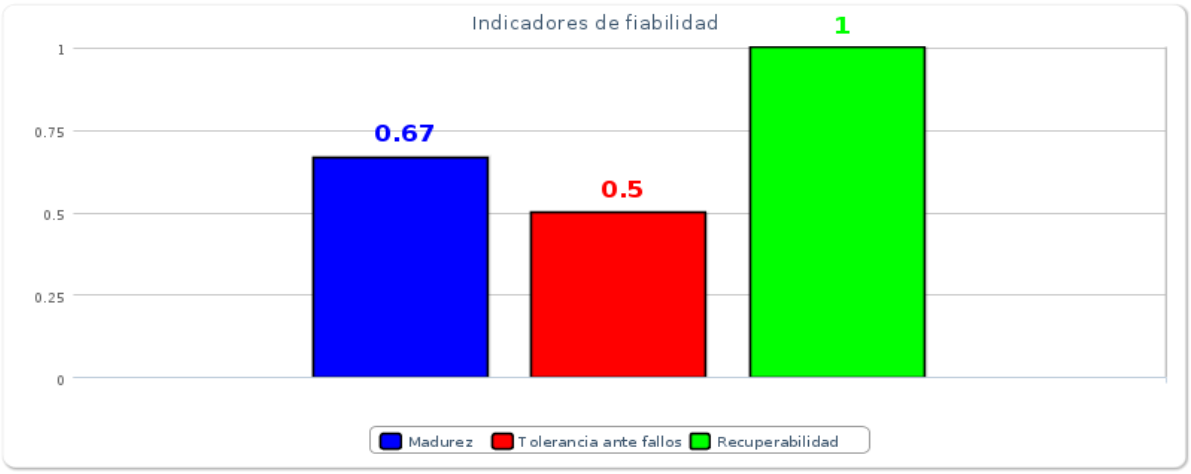

Fig. 7: Indicadores de fiabilidad del componente Gráficas.

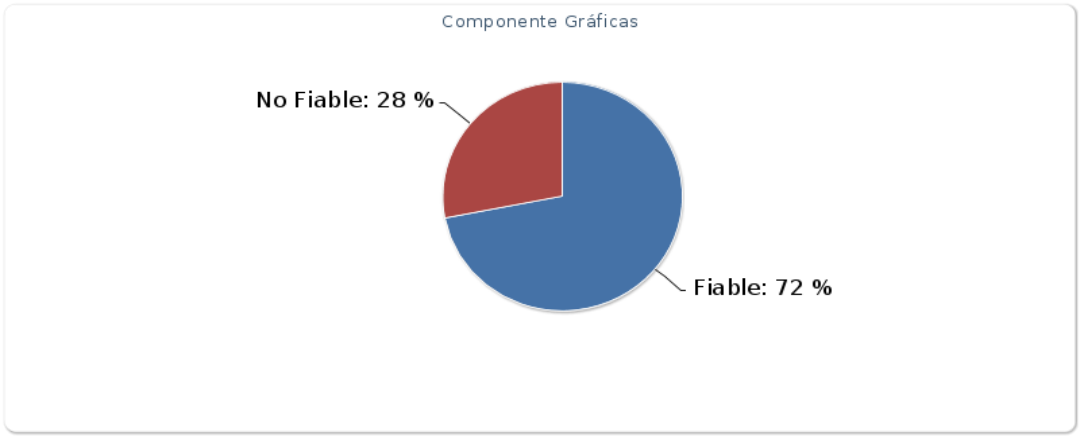

Fig. 8: Por ciento de fiabilidad del componente Gráficas.

3.5 Conclusiones del Capítulo

En este capítulo se realizó la validación del procedimiento propuesto en los componentes “Archivo”, “Gráficas” y “Sonido 2D-3D” los dos primeros pertenecientes al proyecto SIPP y el tercero perteneciente a la línea de productos “Realidad Virtual”. La validación descrita en este capítulo aportó a la investigación una evidencia que confirma que el procedimiento le da solución al problema descrito en la introducción del trabajo. Obteniéndose los valores que posee la fiabilidad de un componente.

Además se permite establecer una comparación de los resultados alcanzados de cada sub-característica en el componente evaluado. Posibilitando al Administrador de la Calidad definir el nivel de fiabilidad que presenta el componente.

También se logra establecer el por ciento de fiabilidad de cada componente, determinando el nivel de calidad del mismo con respecto a la característica evaluada.

CONCLUSIONES GENERALES

Luego de la investigación realizada sobre la evaluación de la fiabilidad y con el fin de fortalecer la calidad de los componentes de software, se le dio cumplimiento al objetivo general de la investigación, al desarrollar un procedimiento que permite evaluar la fiabilidad de los componentes de software que se desarrollan en el CEDIN, logrando un componente con madurez, tolerancia ante fallos y recuperabilidad. Se alcanzó un software con la calidad requerida.

Dicho procedimiento aportó a los administradores de la calidad una guía paso a paso para fortalecer y perfeccionar el desempeño exitoso de este rol, porque permite detectar a tiempo muchos de los defectos que solo eran encontrados en la fase de pruebas internas.

Para obtener dicho resultado se estudiaron un conjunto de modelos y guías que aportaron conocimientos satisfactorios para obtener la propuesta realizada.

Se aplicó el procedimiento en dos componentes que se desarrollaron en el proyecto SIPP, y a un componente de la línea de producto de Realidad Virtual, obteniéndose resultados satisfactorios en uno de los componente evaluados del proyecto SIPP y en el componente evaluado de la línea de productos Realidad Virtual.

RECOMENDACIONES

Para extender la investigación presentada en el presente trabajo de diploma se recomienda:

- Aplicar el procedimiento propuesto a todos los componentes que se desarrollen en el CEDIN para de esta forma contribuir a mejorar la calidad en cuanto a la fiabilidad de los componentes desarrollados o que se encuentren en desarrollo.
- Elaborar una herramienta que permita gestionar el proceso de evaluación de la propuesta realizada.
- Confeccionar un plan de revisiones de evaluación integrada por personal capacitado del GAC del CEDIN, encargada de evaluar la fiabilidad de los componentes de software que se desarrollen.

REFERENCIAS BIBLIOGRÁFICAS

(2010). "*Predicción de Confiabilidad en Sistemas Intensivos en Software*." XII Congreso de Confiabilidad, from http://www.aec.es/c/document_library/get_file?p_l_id=33948&folderId=291408&name=DLFE-7227.pdf.

Boehm, B. W., Kaspar, J.R. y otros. (1978). "*Characteristics of Software Quality* " TRW Series of Software Technology.

Carleton, D. J. P. a. A. D. (1994). "*Case Studies of Software Process-Improvement Measurement*." Computer IEEE 50.

Carrington, R. S. a. P. A. S. a. D. A. (2004). *A Framework for Reliability Assessment of Software Components*. Component-Based Software Engineering, 7th International Symposium, CBSE 2004, Edinburgh, UK, May 24-25, 2004, Proceedings, Springer.

CMMI-SE/SW/IPPD/SS. (2002). *CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing*.

CORPORACION-UNIVERSITARIA-REMINGTON (2010). INGENIERIA DE SISTEMAS CREAD CUCUTA.

Cheung, L. a. R., Roshanak and Medvidovic, Nenad and Golubchik, Leana (2008). *Early prediction of software component reliability*. Proceedings of the 30th international conference on Software engineering, Leipzig, Germany, ACM.

Dill, S. V. A. a. M. J. Z. a. G. (2009). "*A Fast and Robust Reliability Evaluation Algorithm for Generalized Multi-State k -out-of- n Systems*." IEEE Transactions on Reliability **58**: 88-97.

Fernando, H. K. y. R. M. R. (2009). *Análisis de fiabilidad: Los datos y el software de modelización están ayudando a una planta de producción de GNL a determinar métodos de mantenimiento y a mejorar la fiabilidad de los equipos*. Revista ABB, Argentina.

Goaeva-Popstojanova, S. K. a. R. R. L. a. K. (2011). *Empirical evaluation of reliability improvement in an evolving software product line*. Proceedings of the 8th International Working Conference on Mining Software Repositories, MSR 2011 (Co-located with ICSE), Waikiki, Honolulu, HI, USA, May 21-28, 2011, Proceedings, IEEE.

Grady R., C. (1987). *Software metrics: establishing a company wide program*. P. Hall. Nueva Jersey.

Hamlet, D., D. Mason and D. Woit (2001). *Theory of Software Reliability Based on Components*. International Conference on Software Engineering 361–370.

IEEE-Sdt.610.12 (1990). *IEEE Standard Glossary of Software Engineering Terminology*.

IEEE-Std.729 (1983). *IEEE Standard Glossary of Software Engineering Terminology*.

IPP-1000 (2008). *Manual de Procedimientos*.

ISO/IEC-8402 (1986). *Quality - Vocabulary*.

ISO/IEC-9126-1 (2001). *Software engineering -- Product quality -- Part 1: Quality model*.

ISO/IEC-12207 (1995). *Information Technology-Software Life Cycle Processes*.

ISO/IEC-14598 (1999). *Evaluación de los productos de software*.

ISO/IEC-15504-2 (2003). *Information technology -- Process assessment -- Part 2: Performing an assessment*.

ISO/IEC-TR9126-2 (2003). *"Software engineering -- Product quality -- Part 2: External metrics"*.

ISO/IEC-TR9126-3 (2003). *Software engineering -- Product quality -- Part 3: Internal metrics*.

Juan P.C, X. F. y. C. Q. (2009). *Calidad de Componentes Software*.

Li, J., Conradi, R., Slyngstad, O.P.N., Torchiano, M., Morisio, M. y Bunse, C. (2008). "*A State-of-the-Practice Survey of Risk Management in Development with Off-the-Shelf Software Components*." IEEE Transactions on Software Engineering **34**: 2.

Llarena. (2008). "*Metodología para la evaluación de la calidad de estrategias didácticas de cursos a distancia (MACCAD)*." from http://www.citchile.cl/revista-formacion/v1n2fu/art06.pdf.

McCall, J. A., Richards, P.K. and Walters, G.F. (1977). "*Factors in Software Quality*." 86-90.

Montilva, J. A. (2012). "*Desarrollo de Software Basado en Componentes*." from http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf.

Nicanor, L. D. (2003). *Evaluación de la Calidad en Sistemas de Información en Internet*. Departamento de Ingeniería Eléctrica. México, Centro de Investigacion y de Estudios Avanzados del IPN. **Máster:** 102.

Ovaska, M. P. a. A. E. a. E. (2011). "*The reliability estimation, prediction and measuring of component-based software*." Journal of Systems and Software **84**: 1054-1070.

Pressman, R. S. (1998). *Ingeniería del Software, un enfoque práctico*, Mc Graw Hill.

Rantanen, R. D. S. a. E. M. (2007). *Evaluation of a Software Implementation of the Cognitive Reliability and Error Analysis Method (CREAM)*. Pocedings of the Human Factors And Ergonomics Society 51st Annual Meeting.

Reza, V. R. a. J. (2011). "*A Bayesian Based Model for Evaluation of Software Architecture Reliability* " European Journal of Scientific Research **63**(EuroJournals Publishing, Inc. 2011): 243-254.

Roshandel, R. a. B., Somo and Cheung, Leslie and Medvidovic, Nenad and Golubchik, Leana (2006). *Estimating software component reliability by leveraging architectural models*. ICSE, ACM.

Roshandel, R. a. M., Nenad and Golubchik, Leana (2007). *A Bayesian model for predicting reliability of software systems at the architectural level*. Proceedings of the Quality of software architectures 3rd international conference on Software architectures, Springer-Verlag.

Sommerville, I. (2005). Ingeniería del Software

Vergilio, E. O. C. a. A. P. a. S. R. (2010). "*A Genetic Programming Approach for Software Reliability Modeling*." IEEE Transactions on Reliability **59**: 222-230.

Wang, Z. a. W., Jinde and Liang, Xue (2007). "*Non-parametric estimation for NHPP software reliability models*." Journal of Applied Statistics **34**: 107-119.

ANEXOS

Anexo 1

Entrevista a realizar al líder de proyecto al que pertenece el componente a evaluar.

Esta entrevista tiene como objetivo recopilar toda la información inicial necesaria para evaluar la fiabilidad de los componentes que se desarrollan en el CEDIN.

Preguntas

Nombre del componente de software:_____

Tipo de componente de software:

___ Orientado a Objetos.___ Imperativos.___ Otros

Estado del componente de software

___ Terminado___ En desarrollo.

¿Qué Modelo de Calidad se utiliza en el proyecto?

_____

¿En qué fase de desarrollo se encuentra el componente?

_____

Anexo 2

Ciclo de vida de proyectos que se encuentran inmersos en el Programa de Mejora

Estudio Preliminar
Modelación del Negocio
Requisitos
Análisis y Diseño
Implementación

Pruebas Internas
Pruebas de Liberación
Despliegue
Soporte

Anexo 3

Especificaciones de la Tabla 1 Sub- Características de la fiabilidad en las fases del desarrollo de software según los roles.

Roles: se muestran los roles que tienen los permisos pertinentes para evaluar la fiabilidad en un componente de software.

Sub-Características: Nombre del atributo de la fiabilidad a evaluar.

Nivel Requerido: Peso de las características permite a los evaluadores establecer criterios de comparación con el Nivel Real (ver tabla 2).

Fase: Fases del Desarrollo de Software según Programa de Mejora.

Anexo 4

Especificaciones de la Tabla 4 Ejecución de la evaluación:

Sub-características: Colocar nombre de la sub-característica de la fiabilidad a evaluar.

Nivel Requerido: Se especifica el que se encuentra en la Tabla 1 Sub- Características de la fiabilidad en las fases de desarrollo del software según los roles que se encuentra en el epígrafe 2.2.8 Paso 1.

Resultados: Resultado obtenido de la evaluación.

Valor de la métrica: valor del resultado de la métrica aplicada

Nivel Real: nivel que tiene la característica después de haberla evaluado.

Por ciento de fiabilidad: Porciento total de fiabilidad que posee el componente.

Anexo 5

Informe de evaluación de la fiabilidad.

1. **Descripción del componente que se evaluará.**

<Breve descripción del componente que se evaluará>

✓ Consultar Nombre, Tipo, Estado y Fase del componente a evaluar.

<Se consulta el nombre, tipo, estado y fase del componente que se evaluará mediante una entrevista al jefe del proyecto donde se desarrolla el componente>

✓ Identificar posibles errores y proponer acciones correctivas.

<Se identifican los posibles errores de las sub-características [madurez, tolerancia ante fallos, y recuperabilidad] que el componente podría tener y se proponen acciones correctivas para el mejoramiento de las mismas>

✓ Identificar el rol que ejecuta la evaluación

<Se especifica el rol que ejecuta el procedimiento>

✓ Identificar los atributos de fiabilidad a evaluar por fases según el rol

<Se identifican los atributos de fiabilidad que se vayan a evaluar según las necesidades del rol encargado de ejecutar el procedimiento, los atributos son: madurez, tolerancia ante fallos y recuperabilidad>

2. **Especificar métricas**

<Se procede a escoger las métricas necesarias para cada sub-característica, en la siguiente tabla se realiza esta actividad>

Sub-Características	Nombre de las métricas escogidas para realizar la evaluación.

3. Ejecución de la evaluación

<Se procede a ejecutar la evaluación del procedimiento>

Tabla: Ejecución de la evaluación

Sub-característica	Nivel Requerido	Resultados	
		Valor Métrica	Nivel Real
Porciento de fiabilidad			

4. Analizar posibles resultados

<Se analizan los resultados dando las conclusiones de la evaluación efectuada.>

5. Incluir en el Plan de Mediciones las métricas escogidas para realizar la evaluación.