Facultad 5

Sistema para la evaluación de productos informáticos a partir del cálculo de métricas en el Centro de Informática Industrial

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Annabel Columbié Hernández

Tutor: Ing. Lannie Octavio Herrera Pérez

Cotutora: Ing. Lianet Ballester Jiménez

"Año 54 del Triunfo de la Revolución"

La Habana, Cuba

Junio 2012

La posibilidad de realizar un sueño es lo que hace que la vida sea interesante.

Paulo Coelho

Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo a la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Annabel Columbié Hernández

_____

Ing. Lannie Octavio Herrera Pérez

_____

Ing. Lianet Ballester Jiménez

Datos de contacto

Ing. Lannie Octavio Herrera Pérez: Ingeniero en Ciencias Informáticas, graduado del 2007 con Título de Oro y Premio Mella a nivel UCI, en la producción consta de 7 años de experiencia. Profesor Instructor 4 años de experiencia en temas relacionados con aplicaciones automáticas en especial SCADAS, 2 años de experiencia en temas gestión de proyectos, 2 años de experiencia en temas de arquitectura de software.

E-mail: lherrera@uci.cu.

Agradecimientos

Esta tesis representa en mi vida el acontecimiento que marca un antes y un después, la experiencia universitaria, camino recorrido lleno de ganancias y pérdidas; la entrega a un futuro que me invita a soñar y a luchar por mis sueños. En estos años mi vida ha sido bendecida con personas especialmente maravillosas, a las cuales agradezco infinitamente por haber dejado su huella en mí, porque sin su presencia no hubiera sido posible este momento.

A Dios por el regalo de estos 5 años, por cada una de las experiencias vividas y por cada persona con la que he compartido. De manera especial por su fidelidad, por ser el sentido de mi vida y guiar mis pasos, por haberme dado la fuerza para llegar hasta este momento.

A toda mi familia tan grande y unida, por estar siempre atenta a mi formación personal y profesional, por su confianza que para mí es lo más importante. A mis abuelos, mis padres, mis tíos, por tanto amor y dedicación, por sus esfuerzos para que yo llegara a este momento. A mis hermanos y primos por su cariño y por regalarme momentos de alegría.

A la comunidad de las Hijas de la Caridad por el regalo de su presencia en todos estos años, por preocuparse por mi crecimiento personal, profesional y espiritual, por enseñarme a descubrir en los pequeños detalles de la vida la grandeza del amor.

A mis amigos de la iglesia y al padre Luis que aun en los momentos difíciles han hecho y hacen que todo el tiempo sea motivo de alegría y de perseverancia.

A mis amigos de la UCI, gracias por permitirme compartir junto a ustedes, aprender, reír, llorar, por aguantar mis sermones y también mis malcriadeces.

A Yanet que ha sido más que una amiga como una hermana, por aguantarme todos estos años, por aceptarme como soy, por todos los momentos que hemos compartido, por apoyarme siempre, por su amistad incondicional. A su familia por su cariño y acogida.

A Ricardo y Yusmary que en tan poco tiempo se han ganado mi cariño, por su apoyo incondicional.

A todos aquellos profesores que durante estos 5 años han pasado por mi vida dejando sus enseñanzas y de manera especial a aquellos que nunca han dejado de estar, que han sido profesores y amigos.

A mi tutor y mi cotutora por su apoyo, paciencia y confianza en mí, por su exigencia y crítica en la elaboración de este trabajo.

A todas aquellas personas que de una forma u otra han contribuido a la realización de este proyecto.

Dedicatoria

A mis abuelos Pipina y Tony, mi mamá y mi tia Sucel, que lo han dado todo por mí, por su amor, esfuerzo y sacrificio, porque sin su confianza y apoyo no hubiese llegado hasta aquí.

A las Hijas de la Caridad, de manera especial a Sor Marlys y Sor Yubi, porque han sido motivo de inspiración en todo momento, por invitarme siempre a luchar por lo que quiero, por su amistad y presencia en todos estos años.

Resumen

La medición es considerada una parte indispensable de las pruebas de software, es la base para: detectar las desviaciones del rendimiento en los procesos y productos, las oportunidades de mejora, identificar y priorizar las principales preocupaciones, dar seguimiento a la solución y mejorar la calidad del producto

El tema del trabajo surge como respuesta a la necesidad del CEDIN de tener un conocimiento de la calidad de los productos desarrollados en el mismo, es por ello que el objetivo del trabajo es desarrollar una herramienta para el cálculo de métricas en la etapa de pruebas, que facilite al equipo de desarrollo realizar la toma de acciones correctoras en caso de ser necesario, basado en el conocimiento de los indicadores obtenidos.

Se brinda una solución informática a modo de aplicación web que calcula las métricas de software basada en el modelo de la ISO 9126, utilizando como lenguaje de desarrollo PHP y como gestor de base de datos MySQL.

Índice de contenido

Índice de figuras

Introducción:

El mundo actual es un mundo dinámico, donde lo que es válido hoy, quizás mañana no tenga el mismo valor y la única constante es el cambio mismo; por lo que las nuevas organizaciones requieren actualizar los recursos materiales, y lo más importante, la capacidad humana, a fin de dar respuesta puntual, rentable y efectiva a los nuevos desafíos.

En medio de esta realidad y en respuesta a estas necesidades surgen las Tecnologías de la Información y la Comunicación (TIC), como el camino a seguir en pos de nuestro desarrollo social. El gran auge alcanzado por las TIC debido a la aceptación lograda, ha arrojado como resultado un gran desarrollo en las industrias del software en todo el mundo.

Nuestro país de manera específica ha descubierto en las TIC una vía para el desarrollo económico, social y cultural, motivo por el cual se decidió emprender un proceso de informatización en nuestra sociedad, el mismo tuvo como base la creación de la Universidad de las Ciencias Informáticas (UCI).

La UCI unida a la empresa ALBET como industria cubana del software constituye ya una fuente importante de ingresos para el país, destacándose en la producción de software con diversos fines, motivo por el cual se ha estructurado por centros especializados en las diferentes áreas de la industria de nuestro país.

El Centro de Informática Industrial (CEDIN) tiene como misión generar soluciones para la industria, tecnologías, productos y servicios informáticos, que cumplan con las necesidades y expectativas de los clientes. Este cuenta dentro de su estructura, con el departamento de Dirección de Proyectos, el cual posee a su vez un Grupo de Gestión de Calidad (GGC), que mediante la realización de revisiones al producto brinda servicios de Aseguramiento de la Calidad.

Durante la ejecución de estas revisiones, se deberían realizar mediciones a través de la utilización de métricas. Métrica es el término que describe muchos y muy variados casos de medición. Siendo esta una medida estadística que se aplica a todos los aspectos de calidad de software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcional, documentación, métodos, proceso, usuario, entre otros. (1)

El GGC ha tenido problemas para evaluar los productos desarrollados en el centro en el período de revisiones sistemáticas, pues no se calculan los atributos de calidad, lo que imposibilita un conocimiento detallado de la calidad y la toma de acciones correctoras en caso de ser necesario, provocando la

disminución de la robustez, flexibilidad, escalabilidad y mantenibilidad de las aplicaciones. La información referente a las métricas y los procedimientos para su evaluación se encuentran disponibles para consultar en formato digital, sin embargo realizar este proceso de forma manual resulta engorroso y poco confiable al existir errores en el cálculo, además no permite generar reporte de las revisiones.

La situación antes expuesta lleva al siguiente **problema a resolver**: ¿Cómo contribuir a la evaluación de productos informáticos a partir del cálculo de métricas en el GGC?

Se plantea que el **objetivo general** de la investigación es desarrollar una herramienta para el cálculo de métricas en el GGC del CEDIN.

Por tanto se define como **objeto de estudio** las métricas de calidad de software.

El **campo de acción** se centra en una herramienta para el cálculo de las métricas de calidad de software en el GGC del CEDIN.

Para dar solución al objetivo general se definen las siguientes **tareas de investigación**:

- ✓ Estudio de las métricas de calidad, atendiendo al estado del arte y las condiciones propias del centro.
- ✓ Estudio de las metodologías de desarrollo de software para realizar el análisis y el diseño de la aplicación.
- ✓ Estudio de las tecnologías y herramientas a utilizar en la implementación del sistema.
- ✓ Análisis y diseño de la aplicación para el cálculo de las métricas en la etapa de pruebas.
- ✓ Implementación del sistema.
- ✓ Realización de las pruebas de software para validar la solución propuesta.

Este documento contiene cuatro capítulos donde se escribe todo el proceso por el que pasa la investigación.

El primer capítulo **Fundamentación Teórica** contiene los conceptos y fundamentos necesarios para el dominio del problema.

El segundo capítulo **Soluciones Técnicas** propone las soluciones necesarias para el desarrollo de la aplicación.

El tercer capítulo **Características del Sistema** realiza el análisis del negocio, definiendo los actores, trabajadores, requerimientos y los casos de uso.

El cuarto capítulo **Diseño del Sistema** define la estructura de la aplicación mediante el modelo de diseño, diagrama de interacción del diseño y diseño de la base de datos.

El quinto capítulo **Implementación y Prueba** contiene la elaboración del sistema y el modelo de prueba del software para garantizar el cumplimiento de los requisitos del cliente.

Los métodos de investigación científico que se utilizan en esta investigación son:

Métodos teóricos:

- ✓ Analítico Sintético: se analizaron documentos, sitios web, artículos electrónicos y libros, buscando las características de los diferentes modelos y herramientas existentes para el cálculo de las métricas de calidad, permitiendo la extracción de los elementos más importantes que se relacionan con la gestión de las métricas de calidad en los proyectos productivos.
- ✓ Análisis Histórico Lógico: se realizó el análisis histórico lógico de los procesos para el cálculo de las métricas de calidad que actualmente se llevan a cabo en los diferentes proyectos productivos en el Laboratorio Industrial de Pruebas de Software.

Métodos empíricos:

- ✓ Entrevistas: se realizaron entrevistas a varios especialistas del laboratorio Industrial de Pruebas de Software, para conocer cómo se desarrolla el proceso de gestión de métricas de calidad en los proyectos productivos (métodos y herramientas).

Capítulo 1 Fundamentación Teórica

El presente capítulo está relacionado con la necesidad de implementar un sistema informático para el cálculo de las métricas de calidad en el GGC del CEDIN, con el objetivo de tener conocimiento de la calidad de los productos desarrollados por el centro y en función de este conocimiento realizar la toma de decisiones necesarias para lograr una calidad óptima.

En este capítulo se abordan los principales procesos de medición, sus características y definiciones, se describen las métricas de software y los distintos niveles organizacionales, así como los estándares y modelos que se utilizan en la implementación de las métricas.

1.1 Calidad de Software.

En la actualidad no se puede hablar del desarrollo alcanzado por empresas y organizaciones sin antes mencionar la presencia de sistemas informáticos en las mismas, el auge alcanzado por estos sistemas ha acelerado en gran medida el desarrollo de software, por este motivo se hace cada vez más necesario que todo software desarrollado tenga la mayor calidad posible.

La calidad de software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente. (1)

1.2 Métricas de Software.

Para controlar la calidad del software es necesario, ante todo, definir los parámetros, indicadores o criterios de medición. En la ingeniería de software se utilizan las mediciones para evaluar la calidad en los resultados del proyecto o producto y para ayudar a tomar las decisiones más acertadas para contribuir a su mejora.

En ocasiones los términos medida, medición y métrica, son usados en el mismo contexto y bajo el mismo significado, pero es necesario aclarar para una mejor comprensión del problema que cada uno tiene su propio significado.

- ✓ La medida proporciona una indicación cuantitativa de la cantidad, dimensiones o tamaño de algunos atributos de un producto.

- ✓ La medición es el acto de determinar una medida.
- ✓ La métrica es una medida del grado en que un sistema, componente o proceso posee un atributo dado.

Las métricas son la maduración de una disciplina que van a ayudar a la evaluación de los modelos de análisis y de diseño, proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y ayudarán en el diseño de pruebas más efectivas.

Es por eso que propone un proceso de medición, el cual se puede caracterizar por cinco actividades:

- ✓ Formulación: La obtención de medidas y métricas del software apropiadas para la representación de software en cuestión.
- ✓ Colección: El mecanismo empleado para acumular datos necesarios para obtener las métricas formuladas.
- ✓ Análisis: El cálculo de las métricas y la aplicación de herramientas matemáticas.
- ✓ Interpretación: La evaluación de los resultados de las métricas en un esfuerzo por conseguir una visión interna de la calidad de la representación.
- ✓ Realimentación: Recomendaciones obtenidas de la interpretación de métricas técnicas trasmitidas al equipo de software. (2)

Las métricas de software consisten en la aplicación continua de mediciones en el proceso de desarrollo del software y sus productos, para proporcionar información relevante a tiempo, mejorando de esta manera tanto los procesos como el producto.

1.2.1 Clasificación de las métricas de software.

Existen numerosas métricas con propósitos diferentes que reflejan o describen las características y particularidades del software, estas miden entre otros aspectos la competencia, desempeño, complejidad y hasta la calidad del software favoreciendo establecer de una manera sistemática y objetiva una visión interna del trabajo mejorando así la calidad del producto. Las métricas de software se clasifican según el contexto en que se aplican en métricas de procesos, de proyectos y de producto.

- ✓ Las **métricas de proceso** se recopilan de todos los proyectos y durante un largo periodo de tiempo. Se caracterizan por: control y ejecución del proyecto y medición de tiempos de las fases.
- ✓ Las **métricas de proyectos** permiten evaluar el estado del proyecto y seguir la pista de los riesgos.

- ✓ Las **métricas de productos** se centran en las características del software y no en cómo fue producido, donde también son productos los artefactos, documentos, modelos, y componentes que conforman el software y se miden aspectos como el tamaño, la calidad, la totalidad, la volatilidad, y el esfuerzo. (3)

Las métricas orientadas al producto son las que constituyen el objetivo de esta investigación.

A continuación se muestra una breve clasificación de métricas de software descritas por Lem O. Ejiogu:

- ✓ **Métricas de complejidad:** Son todas las métricas de software que definen de una u otra forma la medición de la complejidad; Tales como volumen, tamaño, anidaciones, costo (estimación), agregación, configuración, y flujo. Estas son los puntos críticos de la concepción, viabilidad, análisis, y diseño de software.
- ✓ **Métricas de calidad:** Son todas las métricas de software que definen de una u otra forma la calidad del software; Tales como exactitud, estructuración o modularidad, pruebas, mantenimiento, reusabilidad, cohesión del módulo, acoplamiento del módulo, etc. Estas son los puntos críticos en el diseño, codificación, pruebas y mantenimiento.
- ✓ **Métricas de competencia:** Son todas las métricas que intentan valorar o medir las actividades de productividad de los programadores o practicantes con respecto a su certeza, rapidez, eficiencia y competencia. No se ha alcanzado mucho en esta área, a pesar de la intensa investigación académica.
- ✓ **Métricas de desempeño:** Corresponden a las métricas que miden la conducta de módulos y sistemas de un software, bajo la supervisión del sistema operativo o hardware. Generalmente tienen que ver con la eficiencia de ejecución, tiempo, almacenamiento, complejidad de algoritmos computacionales, etc.
- ✓ **Métricas estilizadas:** Son las métricas de experimentación y de preferencia; Por ejemplo: estilo de código, identificación, las limitaciones, etc. Pero estas no se deben confundir con las métricas de calidad o complejidad. (4)

Pressman también propone un grupo de métricas divididas en 6 categorías, que son las siguientes:

- ✓ **Métricas técnicas:** Están centradas en las características del software más que en su proceso de desarrollo.
- ✓ **Métricas de calidad:** Tanto del software desarrollado como de la efectividad del proceso de la ingeniería aplicado.

- ✓ **Métricas de productividad:** Referidas al rendimiento del proceso de desarrollo como función del esfuerzo aplicado.
- ✓ **Métricas orientadas al tamaño:** Miden de forma directa el software y el proceso por el cual se desarrolla.
- ✓ **Métricas orientadas a la función:** Se centran en la funcionalidad o utilidad del programa.
- ✓ **Métricas orientadas a la persona:** Aportan información sobre la forma en que la persona desarrolla software. (5)

Como se puede observar en esta clasificación según Pressman, se encuentran las métricas que constituyen uno de los objetivos de esta investigación, las métricas de software orientadas a la calidad.

1.3 Estándares, modelos y metodología para el desarrollo de métricas.

Un estándar es el documento aprobado por consenso por un organismo reconocido, que proporciona reglas, pautas y/o características para uso común, con el objetivo de obtener un óptimo nivel de resultados en un contexto dado. (6)

Por tanto los estándares son usados en los disímiles aspectos de nuestra vida cotidiana, se basan en la experiencia real y son probados en la práctica, donde abarcan todos los aspectos relacionados con la información y su gestión.

1.3.1 Modelos de calidad que contienen métricas.

El principal objetivo de los ingenieros de software es producir sistemas, aplicaciones o productos de alta calidad, para garantizar el cumplimiento de este objetivo es necesario realizar evaluaciones mediante la utilización de medidas técnicas, que evalúen la calidad de manera objetiva. Para dar respuesta a esta necesidad aparecen los modelos de calidad que contienen métricas.

Un modelo de calidad es, por lo tanto, un conjunto de prácticas vinculadas a los procesos de gestión y el desarrollo de proyectos. Este modelo supone una planificación para alcanzar un impacto estratégico, cumpliendo con los objetivos fijados en lo referente a la calidad del producto o servicio.

1.3.1.1 Modelo de MCCALL (1977)

Describe la calidad como un concepto elaborado mediante relaciones jerárquicas entre factores de calidad, en base a criterios. Los factores de calidad se concentran en tres aspectos importantes de un producto de software: características operativas, capacidad de cambios y adaptabilidad a nuevos

entornos. Las métricas desarrolladas están relacionadas con los factores de calidad y la relación que se establece se mide en función del grado de cumplimiento de los criterios. (3)

Factor	Criterio
Correctitud	Rastreabilidad Completitud Consistencia
Confiabilidad	Consistencia Exactitud Tolerancia ante fallas
Eficiencia	Eficiencia de ejecución Eficiencia de almacenamiento
Integridad	Control de acceso Auditoría de acceso
Usabilidad	Operabilidad Entrenamiento Comunicación
Interoperabilidad	Modularidad Similitud de comunicación Similitud de datos
Mantenibilidad	Simplicidad Concreción
Capacidad de prueba	Simplicidad Instrumentación Auto-descriptividad Modularidad
Flexibilidad	Auto-descriptividad Capacidad de expansión Generalidad Modularidad
Portabilidad	Auto-descriptividad Independencia del sistema Independencia de máquina
Reusabilidad	Auto-descriptividad Generalidad Modularidad Independencia del sistema Independencia de máquina

Tabla 1 Modelo MCCALL

1.3.1.2 Modelo de FURPS (1987)

Modelo desarrollado por Hewlett-Packard (HP) en 1987, abordando un conjunto de factores de calidad de software y sus respectivos atributos, está basado en el modelo de MCCALL. Se utilizan para establecer métricas de la calidad para todas las actividades del proceso de desarrollo de un software, inclusive de un sistema de información. (3)

Factor	Criterio
Funcionalidad	Características y capacidades del programa Generalidad de las funciones Seguridad del sistema
Facilidad de uso	Factores humanos Factores estéticos Consistencia de la interfaz Documentación
Confiabilidad	Frecuencia y severidad de las fallas Exactitud de las salidas Tiempo medio de fallos Capacidad de recuperación ante fallas Capacidad de predicción
Rendimiento	Velocidad de procesamiento Tiempo de respuesta Consumo de recursos Rendimiento efectivo total Eficacia
Capacidad de soporte	Extensibilidad Adaptabilidad Capacidad de pruebas Capacidad de configuración Compatibilidad Requisitos de instalación

Tabla 2 Modelo FURPS

1.3.1.3 Modelo de DROMEY (1996)

Resalta el hecho de que la calidad del producto es altamente determinada por los componentes del mismo. Sugiere el uso de cuatro categorías que implican propiedades de calidad, que son: correctitud, internas, contextuales y descriptivas. (3)

Factor	Criterio
Correctitud	Funcionalidad Confiabilidad
Internas	Mantenibilidad Eficiencia Confiabilidad
Contextuales	Mantenibilidad Reusabilidad Portabilidad Confiabilidad
Descriptivas	Mantenibilidad Reusabilidad Portabilidad Usabilidad

Tabla 3 Modelo DROMEY

1.3.1.4 Modelo MOSCA (Modelo Sistémico de Calidad)

Consta de 4 niveles: dimensiones, categorías, características y las métricas. En base de tres ramas: el producto, el proceso y la humana. Contiene un total de 715 métricas.

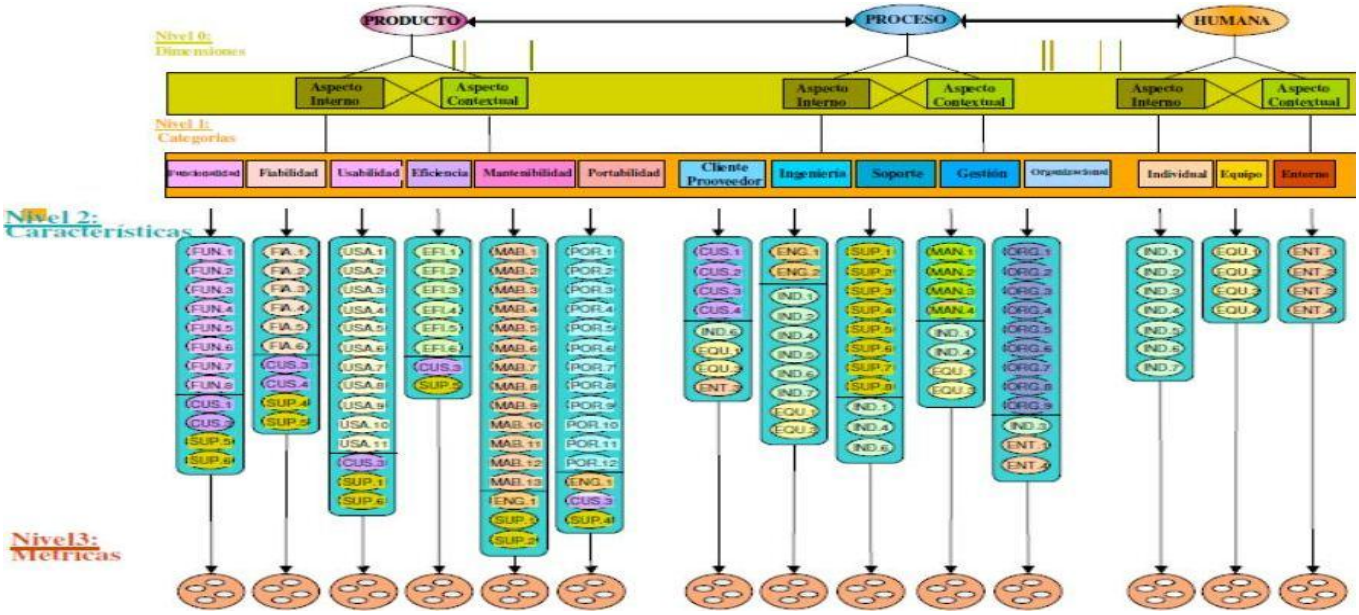


Figura 1 Modelo MOSCA

1.3.1.5 Modelo ISO/IEC 14598

La ISO/IEC 14598 Evaluación de los productos software 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 compone de 6 partes, ellas son: Panorama general, Planificación y gestión, Proceso para desarrolladores, Proceso para adquirientes, Proceso para evaluadores y Documentación de los módulos de evaluación. Esta norma es para utilizarla en conjunto con la 9126 debido a que la 14598 define el proceso de evaluación pero la 9126 tiene el modelo de calidad y las métricas necesarias para llevar a cabo el proceso.

1.3.1.6 Modelo ISO 9126.

Este describe para la calidad de los productos de software dos partes: a) calidad externa y calidad interna, b) calidad durante el uso.

Se especifican seis características para la calidad interna y externa, que son además divididas en sub-características que se manifiestan externamente cuando el software se usa como una parte del sistema computarizado, y son un resultado de los atributos o cualidades internos del software.

Las métricas internas le proporcionan a los desarrolladores la habilidad de medir la calidad de estos productos intermedios, con lo cual se puede predecir la calidad del producto final. Esto le permite a los desarrolladores identificar los problemas que afecten la calidad e iniciar las acciones correctivas en las etapas tempranas del ciclo de vida de desarrollo del producto. (7)

Las métricas externas pueden ser usadas para medir la calidad del producto de software a través de la medición del comportamiento del sistema del cual el software forma parte.

Cada una de estas características de calidad interna y externa, evalúa otras subcaracterísticas que se derivan de ellas, como se puede observar en la siguiente figura:

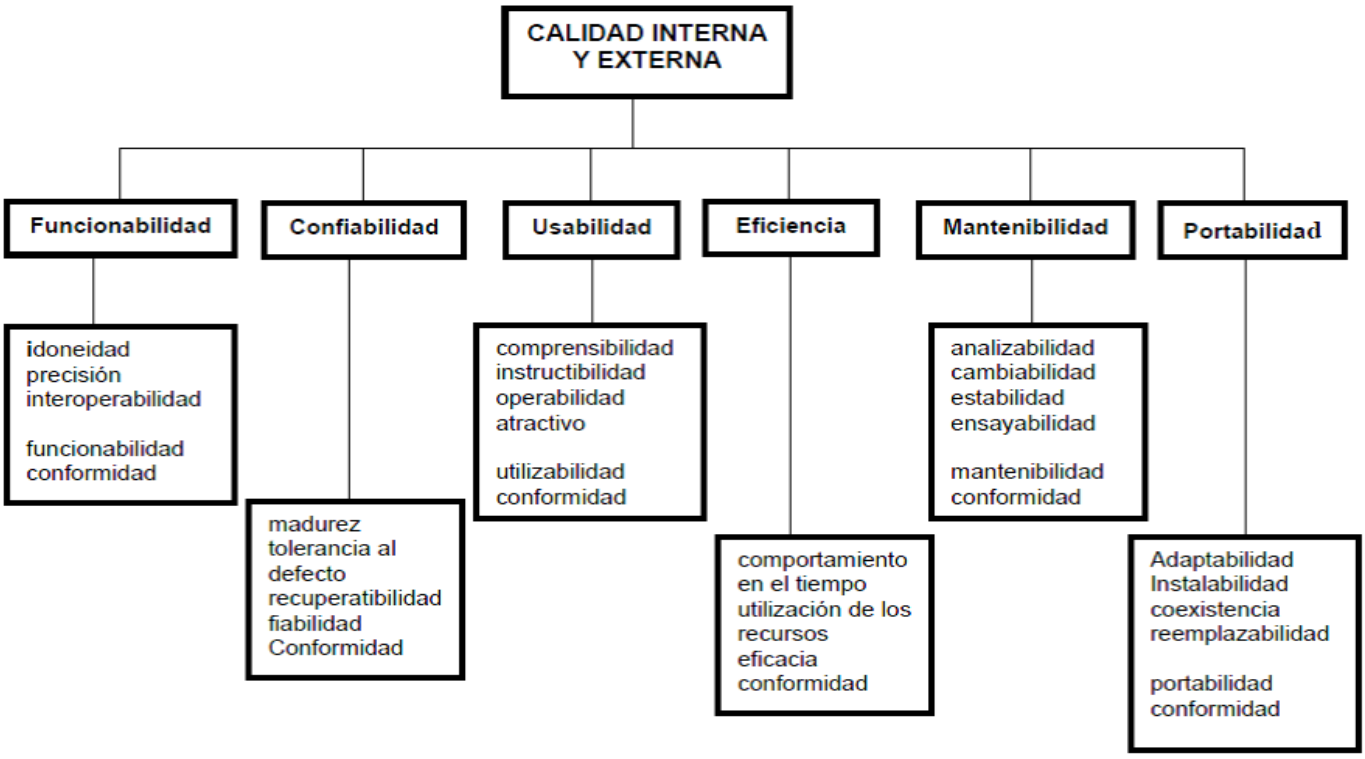


Figura 2 Atributos de Calidad Interna/Externa ISO 9126

Para la calidad durante el uso se definen cuatro características. La calidad durante el uso es el efecto combinado que percibe el usuario de la calidad interna y externa del software. (8)

Las métricas de calidad en uso miden si un producto resuelve las necesidades de usuarios específicos para alcanzar metas específicas con eficacia, productividad, seguridad y satisfacción en un contexto específico de uso. Esto solo puede lograrse en un entorno real del sistema.

En cuanto a los factores de calidad de uso ISO 9126 reconoce cuatro factores:

- ✓ Eficacia.
- ✓ Productividad.
- ✓ Seguridad.
- ✓ Satisfacción.

1.3.2 Metodologías para el desarrollo de métricas de calidad.

En la actualidad la calidad constituye uno de los elementos diferenciadores en el ámbito mundial entre las compañías desarrolladoras de software, por esta razón la necesidad de la calidad en los sistemas ha propiciado la creación de modelos, frameworks y metodologías para evaluar y asegurar su calidad.

Teniendo en cuenta que los diferentes usuarios que interactúen con el software pueden tener diferentes puntos de vista de lo que significa calidad, se debe determinar qué medir haciendo uso de las metodologías de calidad y modelos de calidad bien definidos.

Para el proceso de definición de las métricas se tiene que empezar por documentar el proceso de desarrollo mediante recopilación de datos identificados, seguido del establecimiento de las métricas orientadas a alcanzar la calidad requerida. Una vez definidas las métricas se hace necesaria la selección de las herramientas para el análisis de estas y por último la creación de una base de datos para almacenar la información recolectada, incluyendo las métricas definidas.

1.3.2.1 IEEE 1061-1998 Standard for a Software Quality Metrics Methodology.

Denominado Metodología para Métricas de Calidad del Software (*IEEE Standard for a Software Quality Metrics Methodology*), este estándar provee una metodología para el establecimiento de requerimientos de calidad e identificar, implementar, analizar y validar métricas de calidad de software para procesos y productos. Es aplicable a cualquier software en todas las fases de su ciclo de vida. Está dirigido a aquellos que se encuentran asociados a la adquisición, desarrollo, uso, soporte, mantenimiento y auditoría de software. Puede ser empleado por: gerentes de proyectos, desarrolladores, auditores y usuarios.

Esta metodología permite a los que la utilizan:

- ✓ Evaluar el logro de las metas de calidad.
- ✓ Establecer los requerimientos de calidad para un sistema a su salida.
- ✓ Establecer estándares y criterios de aceptación.
- ✓ Evaluar el nivel de calidad alcanzado contra los requerimientos requeridos.
- ✓ Detectar anomalías o puntos de problemas potenciales en el sistema.
- ✓ Predecir el nivel de calidad que será alcanzado en el futuro.
- ✓ Monitorear los cambios en la calidad del software si este es modificado.
- ✓ Evaluar la facilidad de cambio en el sistema durante la evolución del producto.

El estándar comprende cinco pasos que deben realizarse de manera iterativa, ya que los resultados de uno de ellos pueden necesitarse en pasos subsiguientes. El conjunto de tareas a llevar a cabo es el siguiente:

1. Establecer los requisitos de calidad del software. Para lo cual es necesario llevar cabo las siguientes tareas:

a) Elaborar un listado con los posibles requisitos de calidad, utilizando para ello toda la información disponible: estándares, requisitos de los contratos o compras (tales como garantías o fechas de entrega).

b) Identificar la lista de requisitos de calidad a partir de la lista de posibles requisitos del paso anterior. Dicha lista se obtendrá evaluando la importancia de cada uno y eliminando posibles contradicciones.

c) Cuantificar los factores de calidad. A cada factor de calidad se le debe asignar al menos una métrica con sus respectivos valores.

Por ejemplo, si un requisito fundamental es la *habilidad* en un sistema Web, una métrica podría ser la "Disponibilidad diaria" con un valor de 95 %, o lo que es lo mismo disponibilidad de 1.368 minutos sobre los 1.440 minutos que componen las 24 horas de un día.

2. Identificar las métricas de calidad del software mediante la aplicación de un marco de métricas de calidad, la realización de un análisis sobre costes y beneficios y el establecimiento de las garantías necesarias para implantar dichas métricas.

3. Implementar las métricas de calidad. En este paso se definen los procedimientos de medición, se hace un prototipo del proceso de medición y finalmente se lleva a cabo la medición en sí.

4. Analizar los resultados. Consiste en interpretarlos, analizando si los requerimientos de calidad se ajustan a los requisitos definidos en la especificación.

5. Evaluar los resultados. Para ello se aplican criterios y metodologías de evaluación.

El estándar no prescribe, de manera explícita, ninguna métrica específica, limitándose a facilitar y promover el uso de métricas dentro de las organizaciones y en el contexto de proyectos con el objetivo final de mejorar la calidad del software producido. (9)

1.3.2.2 Método Objetivo – Pregunta - Métrica (GQM)

El método Objetivo-Pregunta-Métrica, o GQM *(Goal-Question-Metric*) como es más conocido, fue desarrollado para alcanzar los objetivos de calidad requeridos por la NASA en los años 70. GQM ayuda a

identificar, centrar, documentar y analizar un número reducido de métricas con la intención de mejorar un objetivo que puede ser tanto del producto como del proceso o sus recursos. El método se basa en tres niveles:

- ✓ El primero es el nivel conceptual, en el que se identifica un objetivo de calidad. Dicho objetivo, que puede estar relacionado con la evaluación o la mejora, será el propósito de la medición en relación a una entidad –producto, proceso o recurso– y desde un punto de vista específico (gestor, desarrollador, operador, mantenimiento, etc.).
- ✓ El segundo nivel, llamado nivel operacional, divide el objetivo en una serie de preguntas que caracterizan a la entidad.
- ✓ Finalmente, el nivel cuantitativo especifica el conjunto de métricas necesarias para poder responder a las preguntas planteadas en el Segundo.

El funcionamiento del método se basa en el refinamiento progresivo de un conjunto de objetivos de negocio *(G-Goals*) que se establecen como partida. Tomando dichos objetivos como entrada y mediante el planteamiento de preguntas *(Q- Questions),* se obtienen finalmente un conjunto de métricas *(M-Metrics)* específicas que permitirán medir los objetivos enunciados. (9)

1.4 Herramientas para el cálculo de métricas.

Existen diferentes herramientas en el mundo y también en nuestro país para el cálculo de métricas, que contribuyen al conocimiento y retroalimentación de la calidad de software, pero ninguna se adapta a las necesidades del centro.

Entre las herramientas existentes en el mundo se pueden mencionar:

- ✓ **Metrics:** Es una herramienta para el cálculo de métricas, de producto de software, a través de la medición y evaluación del código fuente. Permite medir distintas características como la cohesión, la complejidad y otros atributos interesantes tanto de los paquetes, de las clases y sus métodos. Muchas de estas métricas y sus resultados pueden transformarse en información básica para el cálculo de otras métricas más complejas. La herramienta se vuelve importante al automatizar la medición de atributos que serían muy tediosos de medir en forma manual. Metrics es un plugin de Eclipse que se integra a la IDE de desarrollo. Permite configurar las métricas que se quieren recolectar y definir sus tolerancias. Esto facilita la ejecución y seguimiento de las métricas que interesan; y advertir los resultados que están fuera de rango para poder tomar acciones correctivas. (10)

- ✓ **SourceMonitor:** es una herramienta para el cálculo de las métricas, desarrollado por Campwood Software en Burlington, es una empresa privada, se estableció en 1997 e incorporada en Vermont calificados en Servicios de programación de computadoras. Empresas como Software Campwood por lo general ofrecen: Programación Avanzada, Programa del Ejército de equipos pequeños, Programación básica. (11)

También en la UCI existen herramientas de este tipo:

- ✓ **Sistema Calculador de Métricas para evaluar la calidad de un software en el ERP:** Brinda una solución informática a modo de aplicación web, que evalúa la calidad de software a través del cálculo de métricas. Este sistema gestiona los atributos de calidad, las métricas de software y las escalas, que son necesarios para realizar la evaluación. Además, el mismo gestiona configuraciones mediante las cuales realiza la evaluación y genera un reporte con el resultado de la misma. La aplicación fue desarrollada a raíz de la necesidad por parte del Grupo de Calidad del proyecto ERP- Cuba, de contar con una herramienta que le permita mantenerse informado constantemente sobre el estado de los productos que en el proyecto se desarrollan.

1.5 Conclusiones del capítulo.

El estudio de las métricas, normas y metodologías realizado en este capítulo, permitió adquirir el conocimiento necesario para la identificación y posterior selección de la metodología de desarrollo de métricas de calidad y el modelo más adecuado para implementar en la aplicación. Propició además un mejor entendimiento del proceso de la evaluación de las mismas.

Capítulo 2 Soluciones Técnicas

En el presente capítulo se proponen las soluciones técnicas para crear una aplicación que permita calcular de manera automatizada los diferentes tipos de métricas.

2.1 Metodología para el desarrollo de métricas de calidad: IEEE Standard for a Software Quality Metrics Methodology.

Se seleccionó como metodología para el desarrollo de métricas el estándar de la IEEE 1061 del 1998 debido a que esta permite identificar, implementar, analizar y validar que métricas son las adecuadas a desarrollarse en el contexto del proyecto, sin prescribir ninguna métrica en específico. Además propone una serie de pasos detallados que sirven de guía para el desarrollo, abarcando el estudio de los requisitos de calidad, la identificación e implementación de las métricas y posteriormente la evaluación de los resultados.

2.2 Modelo de calidad: ISO 9126.

Como métricas a implementar en la aplicación se decidió utilizar las propuestas por el modelo de calidad de la ISO 9126, debido a que su flexibilidad y facilidad de implementación permite a las empresas que la utilicen adecuarla a su entorno y sus necesidades objetivas, creando su propio modelo de calidad. Aprovechando estas particularidades se seleccionaron de todas las métricas contenidas en este modelo, las que más se adecuan a las necesidades y características del centro.

Con el fin de lograr un entendimiento de estos atributos, se explican a continuación las definiciones de cada una de las características de calidad seleccionadas y sus respectivas subcaracterísticas:

Funcionalidad: Conjunto de atributos que relacionan la existencia de un conjunto de funciones con sus propiedades especificadas. Las funciones satisfacen necesidades especificadas.

- ✓ **Idoneidad:** Atributos que determinan si el conjunto de funciones son apropiadas para tareas especificadas.
- ✓ **Precisión:** Atributos que determinan que los efectos sean los correctos o los esperados.
- ✓ **Seguridad:** Atributos que miden la habilidad para prevenir accesos no autorizados.

Confiabilidad: Conjunto de atributos que se relacionan con la capacidad del software de mantener su nivel de performance bajo las condiciones establecidas por un período de tiempo.

- ✓ **Madurez:** Atributos que se relacionan con la frecuencia de fallas por defectos en el software.

- ✓ **Tolerancia a las fallas:** Atributos que miden la habilidad de mantener el nivel especificado de performance en caso de fallas del software.
- ✓ **Recuperabilidad:** Atributos que miden la capacidad de restablecer el nivel de performance y recuperar datos en caso de falla, y el tiempo y esfuerzo necesario para ello.

Usabilidad: Conjunto de atributos que se relacionan con el esfuerzo necesario para usar, y en la evaluación individual de tal uso, por parte de un conjunto especificado de usuarios.

- ✓ **Comprensibilidad:** Atributos que miden el esfuerzo del usuario en reconocer el concepto lógico del software y su aplicabilidad.
- ✓ **Instructibilidad:** Atributos que miden el esfuerzo del usuario en aprender la aplicación.
- ✓ **Atractivo:** Atributos que miden la capacidad del producto de software de ser amigable para el usuario.

Eficacia: Conjunto de atributos que se relacionan con el nivel de performance del software y la cantidad de recursos usados bajo las condiciones establecidas.

- ✓ **En tiempo:** Atributos que miden la respuesta y tiempos de procesamiento de las funciones.
- ✓ **Rendimiento**: Atributos que miden la cantidad de recursos usados y la duración de tal uso en la ejecución de las funciones.

Portabilidad: Conjunto de atributos que se relacionan con la habilidad del software para ser transferido de un ambiente a otro.

- ✓ **Adaptabilidad:** Atributos que miden la oportunidad de adaptación a diferentes ambientes sin aplicar otras acciones que no sean las previstas para el propósito del software.
- ✓ **Instalabilidad:** Atributos que miden el esfuerzo necesario para instalar el software en el ambiente especificado.

Para una mejor comprensión de cada métrica y lo que propone evaluar, ver anexo#1.

2.3 Metodologías, herramientas y lenguajes de desarrollo a utilizar.

En este epígrafe se nombran y fundamentan las tecnologías, metodologías, herramientas y lenguajes a utilizar en el diseño e implementación de la aplicación.

2.3.1 Solución propuesta: Sitio Web.

Se determinó desarrollar un sitio web que permita a los usuarios conectarse a través de la red desde cualquier ubicación y calcular de forma automatizada las métricas definidas, de manera que para los productos creados en el CEDIN se pueda obtener una evaluación cuantitativa de su calidad.

2.3.2 Lenguaje de programación PHP.

PHP o *Hypertext Pre-processor* es un lenguaje de scripting ampliamente utilizado de código abierto que es especialmente adecuado para el desarrollo web, por este motivo será el que se utilice para el desarrollo del sitio en cuestión.

Se centra principalmente en las secuencias de comandos del lado del servidor, por lo que permite entre otras operaciones, recopilar datos de formularios, generar páginas con contenidos dinámicos, así como enviar y recibir cookies.

Entre las principales ventajas del lenguaje PHP se destacan:

- ✓ Es un lenguaje multiplataforma.
- ✓ Capacidad de expandir su potencial utilizando módulos.
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Permite aplicar técnicas de programación orientada a objetos (POO).
- ✓ Es libre.
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida. (12)

2.3.3 Metodología de desarrollo Proceso Unificado Racional (RUP).

Se propone RUP como metodología de desarrollo debido a que sus características y estructura, permiten un mejor entendimiento del negocio del sistema a desarrollar.

El Proceso Unificado Racional o *Rational Unified Process (RUP)* es un proceso de desarrollo de software que junto al Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP permite una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo).

Como metodología se caracteriza por ser:

- ✓ Iterativo e incremental.

- ✓ Centrado en la arquitectura.
- ✓ Guiado por los casos de uso.

RUP divide el proceso de desarrollo en ciclos, cada ciclo a su vez se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- ✓ **Inicio:** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- ✓ **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.
- ✓ **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios release del producto que han pasado las pruebas. Se ponen estos a consideración de un subconjunto de usuarios.
- ✓ **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

2.3.4 Visual Paradigm

Visual Paradigm es la herramienta que se utilizará para el modelado UML, por las ventajas que ofrece para el modelamiento de sistemas y las facilidades que brinda al usuario a partir de una interfaz amigable y fácil de utilizar. Sin embargo la posibilidad de trabajar en cualquier plataforma es lo que la convierte en la mejor elección.

Visual Paradigm es una herramienta que da soporte al modelado visual con UML, que soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm ofrece:

- ✓ Entorno de creación de diagramas para UML 2.1.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.

- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDE.
- ✓ Disponibilidad en múltiples plataformas.

2.3.5 MySQL

El SGBD que se utilizará será MySQL, por su rapidez y capacidad de almacenar grandes cantidades de datos, además de que consta de un sistema de permisos muy potente y usa el lenguaje SQL estandarizado para el almacenamiento, actualización y acceso a información. MySQL posee una interfaz amena y los comandos para gestionar la base de datos son muy intuitivos (más que los de PostgreSQL) siendo muchos de ellos sentencias SQL. Es importante agregar a estas razones que gestor está bajo la licencia GPL.

Principales características de MySQL:

- ✓ Aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multihilo.
- ✓ Soporta gran cantidad de tipos de datos para las columnas.
- ✓ Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc.)
- ✓ Gran portabilidad entre sistemas.
- ✓ Soporta hasta 32 índices por tabla.
- ✓ Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

Ventajas de MySQL:

- ✓ Multiplataforma.
- ✓ Es de código abierto.
- ✓ Es muy rápido, estable, fiable y fácil de usar.
- ✓ MySQL Server trabaja en entornos cliente/servidor o incrustados
- ✓ Soporte a grandes bases de datos.
- ✓ Cuenta con acceso a las bases de datos de forma simultánea.
- ✓ El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas. (13)

2.3.6 Servidor Apache

Se utilizará Apache como servidor de base de datos por ser multiplataforma y ser una tecnología de código abierto, además de ser compatible con php que será el lenguaje de programación que se utilice para el desarrollo de la aplicación.

Apache es un servidor Web potente y flexible. Es una tecnología gratuita y de código abierto, que proporciona transparencia en todo el proceso de instalación. Es prácticamente universal, por su disponibilidad en multitud de sistemas operativos. Posee una alta configurabilidad en la creación y gestión de logs, de este modo es posible tener un mayor control sobre lo que sucede en el servidor.

La arquitectura del servidor Apache es muy modular, consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web.

Los módulos del Apache se pueden clasificar en tres categorías:

- ✓ **Módulos Base:** poseen las funciones básicas del Apache.
- ✓ **Módulos Multiproceso:** Son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- ✓ **Módulos Adicionales:** Cualquier otro módulo que le añada una funcionalidad al servidor.

2.3.7 PHPStorm

PhpStorm es un Entorno de desarrollo integrado (IDE) ligero e inteligente enfocado en la productividad del desarrollador, este es el IDE que se utilizará para la implementación. PhpStorm provee completamiento inteligente de código, navegación rápida y chequeo de errores al momento. Siempre está listo para dar forma al código, ejecutar unit o proveer debugging visual.

Entre los principales beneficios de su uso se puede mencionar.

Es un editor php inteligente con:

- ✓ Completamiento de código php
- ✓ Detector de código duplicado
- ✓ Mezcla lenguajes (JavaScript, SQL, XML)

Es un editor de JavaScript avanzado:

- ✓ Basado en DOM
- ✓ Navegación de código y búsqueda de usos
- ✓ Debbuger de JavaScript

Es un editor HTML/CSS:

- ✓ Soporta HTML5
- ✓ Validación y arreglo-rápido
- ✓ Muestra estilos aplicados

IDE Ligero:

- ✓ Instalación fácil
- ✓ Multiplataforma
- ✓ Código abierto

Ambiente inteligente:

- ✓ Unit php visual
- ✓ Históricos de cambios locales
- ✓ Php UML

Debugging visual:

- ✓ Puntos de ruptura en php, js, html
- ✓ Inspector de variables
- ✓ Análisis de código Batch (14)

2.4 Conclusiones del capítulo.

En este capítulo se abordaron las soluciones técnicas para el desarrollo del sistema a automatizar y se decidió además la implementación de las métricas de calidad propuestas por la norma ISO 9126, quedando bien definidas las soluciones necesarias para el desarrollo de la aplicación.

Capítulo 3 Características del Sistema

En este capítulo se dará una breve descripción del proceso para la evaluación de proyectos a través del cálculo de las métricas, así como el papel que desempeñan las personas involucradas en este proceso. Se identificarán los requisitos funcionales y no funcionales, los casos de uso con sus descripciones, los diagramas de actividades y el modelo de objeto. Se definirá la propuesta del sistema.

3.1 Modelo del negocio

En el proceso de desarrollo de software resulta útil la creación de modelos que organicen y muestren los detalles importantes de problemas reales que se vinculan con el sistema informático a construir. Estos modelos deben cumplir una serie de propiedades, entre ellas la de ser coherentes y relacionados. Uno de los modelos útiles previo al desarrollo de un software es el modelo del negocio.

Los principales objetivos del modelamiento del negocio son:

- ✓ Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.
- ✓ Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- ✓ Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización. (15)

3.1.1 Actor del negocio

“Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados”. (15)

Figura 3 Actor del negocio

3.1.2 Trabajador del negocio

Un trabajador representa a personas o sistemas (software) dentro del negocio que son las que realizan las actividades que están comprendidas dentro de un caso de uso.

Figura 4 Trabajador del negocio

3.1.3 Caso de uso del negocio

"Un caso de uso del negocio representa a un proceso de negocio, por lo que se corresponde con una secuencia de acciones que producen un resultado observable para ciertos actores del negocio". (15)

Figura 5 Caso de uso del negocio

3.1.4 Diagramas de Casos de Uso del negocio

"Un diagrama de casos de uso del negocio representa gráficamente a los procesos del negocio y su interacción con los actores del negocio." (15)

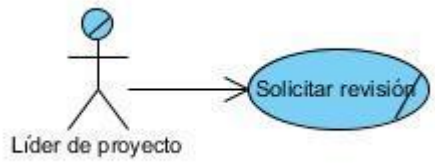


Figura 6 Diagrama de Caso de uso del negocio

3.1.4.1 Especificación de caso de uso del negocio

Nombre	**Solicitar Revisión**
Actor del negocio:	Líder de proyecto.
Propósito:	Evaluar calidad del proyecto mediante el uso de métricas.
Resumen:	El caso de uso inicia cuando el líder de

	proyecto decide evaluar su proyecto.
Flujo Normal de los eventos:	
Acciones del actor	**Respuesta del proceso de negocio**
1- El líder de proyecto llega al laboratorio de calidad y solicita la revisión.	1.1- El especialista de calidad busca los datos del proyecto. 1.2- Selecciona el proyecto. 1.3- Selecciona las métricas. 1.4- Calcula las métricas. 1.5- Obtiene reporte de resultados. **1.6-** Entrega reporte de resultados.
Flujo Alterno **Nº Evento 1. Proyecto no encontrado**	
	1.1 a Solicita adicionar el proyecto
Prioridad:	Crítico.

Tabla 4 Descripción del CU del negocio

3.1.5 Diagrama de actividades

"Los diagramas de actividades ayudan a describir detalladamente que es lo que pasa dentro del negocio. Ellos detallan los roles específicos que juegan las personas (trabajadores del negocio) y las actividades que realizan". (15)

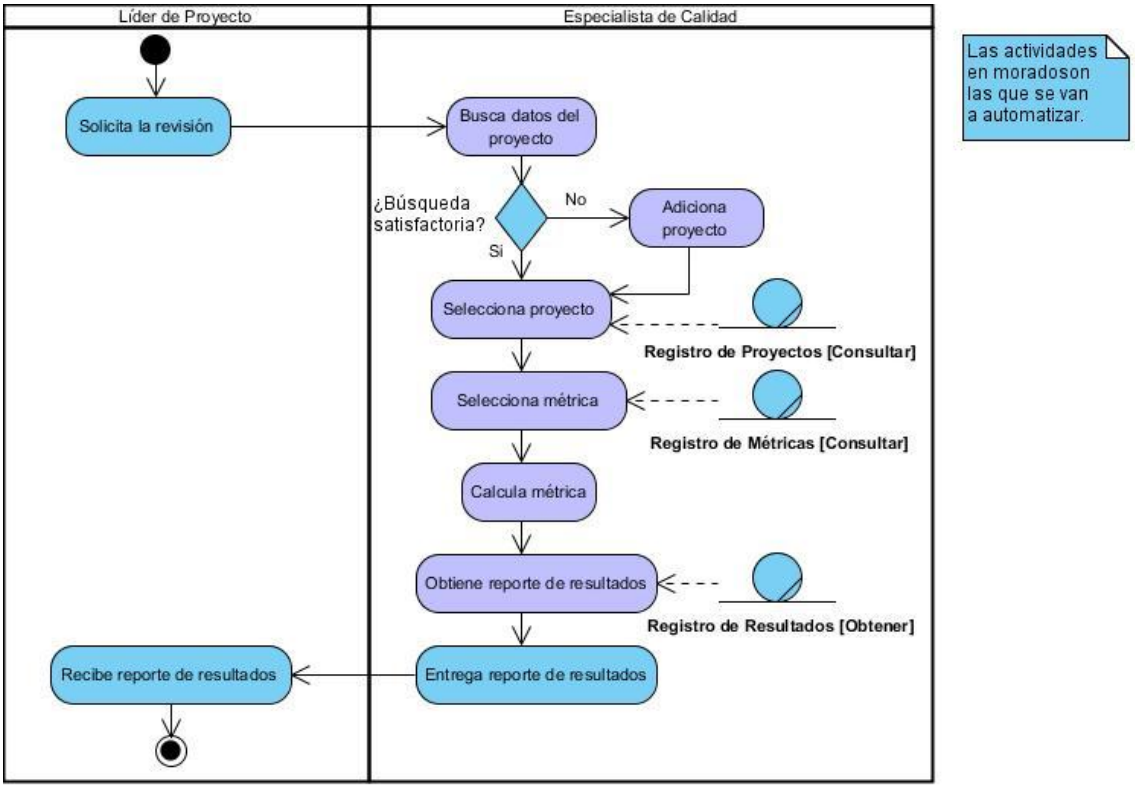


Figura 7 Diagrama de actividades

3.1.6 Diagrama de clases del modelo de objeto

El modelo de objetos del negocio, muestra la participación de los trabajadores y entidades del negocio y la relación entre ellos.

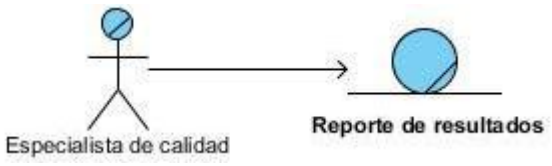


Figura 8 Diagrama de clases del modelo de objeto

3.2 Definición de Requisitos Funcionales

Los requisitos funcionales son aquellas capacidades o funciones que el sistema debe cumplir, teniendo en cuenta las necesidades del usuario. Para lograr satisfacer estas necesidades, se definen los siguientes requerimientos funcionales:

RF1. Autenticar usuario: El sistema permitirá al usuario autenticarse como administrador o especialista de calidad.

RF2. Gestionar proyectos: El sistema permitirá al especialista de la calidad gestionar los proyectos.

RF2.1 Adicionar proyecto: El sistema permitirá al especialista de la calidad adicionar un nuevo proyecto.

RF2.2 Modificar proyecto: El sistema permitirá al especialista de la calidad modificar los datos del proyecto.

RF2.3 Eliminar proyecto: El sistema permitirá al especialista de la calidad eliminar un proyecto.

RF3. Seleccionar proyecto: El sistema permitirá al especialista de la calidad seleccionar el proyecto que desea evaluar.

RF4. Seleccionar métrica: El sistema permitirá al especialista de la calidad seleccionar la característica de la calidad a evaluar. A su vez, cada característica muestra las métricas correspondientes y cada métrica los aspectos a evaluar.

RF5. Mostrar indicaciones para el cálculo de métricas: El sistema debe mostrar al especialista de la calidad las indicaciones para el cálculo de las métricas.

RF6. Calcular métrica: El sistema permitirá al especialista de la calidad calcular la métrica seleccionada según la fórmula establecida en las indicaciones.

RF7. Obtener reporte de resultados: El sistema permitirá al especialista de la calidad obtener el reporte de resultados de la evaluación realizada.

RF8. Gestionar usuarios: El sistema permitirá al jefe de departamento de calidad gestionar los usuarios.

RF8.1 Adicionar usuario: El sistema permitirá al jefe de departamento de calidad adicionar un nuevo usuario.

RF8.2 Modificar usuario: El sistema permitirá al jefe de departamento de calidad modificar los datos del usuario.

RF8.3 Eliminar usuario: El sistema permitirá al jefe de departamento de calidad eliminar un usuario.

RF9. Administrar páginas: El sistema permitirá al jefe de departamento de calidad gestionar las páginas.

RF9.1 Adicionar página: El sistema permitirá al jefe de departamento de calidad adicionar una nueva página.

RF9.2 Eliminar página: El sistema permitirá al jefe de departamento de calidad eliminar una página.

RF10. Gestionar roles de usuarios: El sistema permitirá al jefe de departamento de calidad gestionar los roles de usuarios.

RF10.1 Adicionar usuario: El sistema permitirá al jefe de departamento de calidad adicionar un nuevo rol de usuario.

RF10.2 Modificar usuario: El sistema permitirá al jefe de departamento de calidad modificar los datos de un rol de usuario.

RF10.3 Eliminar usuario: El sistema permitirá al jefe de departamento de calidad eliminar un rol de usuario.

3.3 Definición de Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades son las características que hacen al producto atractivo, usable, rápido o confiable. Los requisitos no funcionales normalmente están vinculados a requisitos funcionales, es decir una vez que se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Apariencia: El sistema será una aplicación web, con una interfaz sencilla y un funcionamiento comprensible para el usuario.

Seguridad: El sistema debe solicitar al usuario que se autentique para poder interactuar con la aplicación. Los usuarios deben ser creados según el rol que desempeñan asignándole los privilegios indispensables.

Usabilidad: El sistema debe ser utilizado solo por los especialistas de la calidad.

Rendimiento: El sistema debe ser robusto y eficiente, garantizando una adecuada capacidad de procesamiento.

Portabilidad: El sistema debe ser multiplataforma.

Software: En el lado del cliente el sistema requiere de un navegador que soporte Adobe Reader y por el lado del servidor debe estar instalado PHP 5.0 o superior, además del gestor de base de datos MySQL.

3.4 Modelo del Sistema

El modelo del sistema permite tener un entendimiento más detallado de cómo va a estar estructurado el sistema a partir de los casos de uso (CU) identificados. Para ello se identifican los actores y se especifican los casos CU del sistema.

3.4.1 Actor del Sistema a Automatizar

Los actores del sistema son agentes externos, es decir, las personas u otros sistemas que interactúan con él.

Actor	Descripción
Usuario	Es el que se autentica para poder acceder al sistema.
Especialista de calidad	Es el que realiza el cálculo de métricas para la evaluación del producto.
Jefe de departamento de calidad	Es el encargado de gestionar los usuarios, los roles y las páginas del sistema y puede además realizar las actividades del especialista de calidad.

Tabla 5 Actores del sistema

3.4.2 Diagrama de Casos de Uso del Sistema.

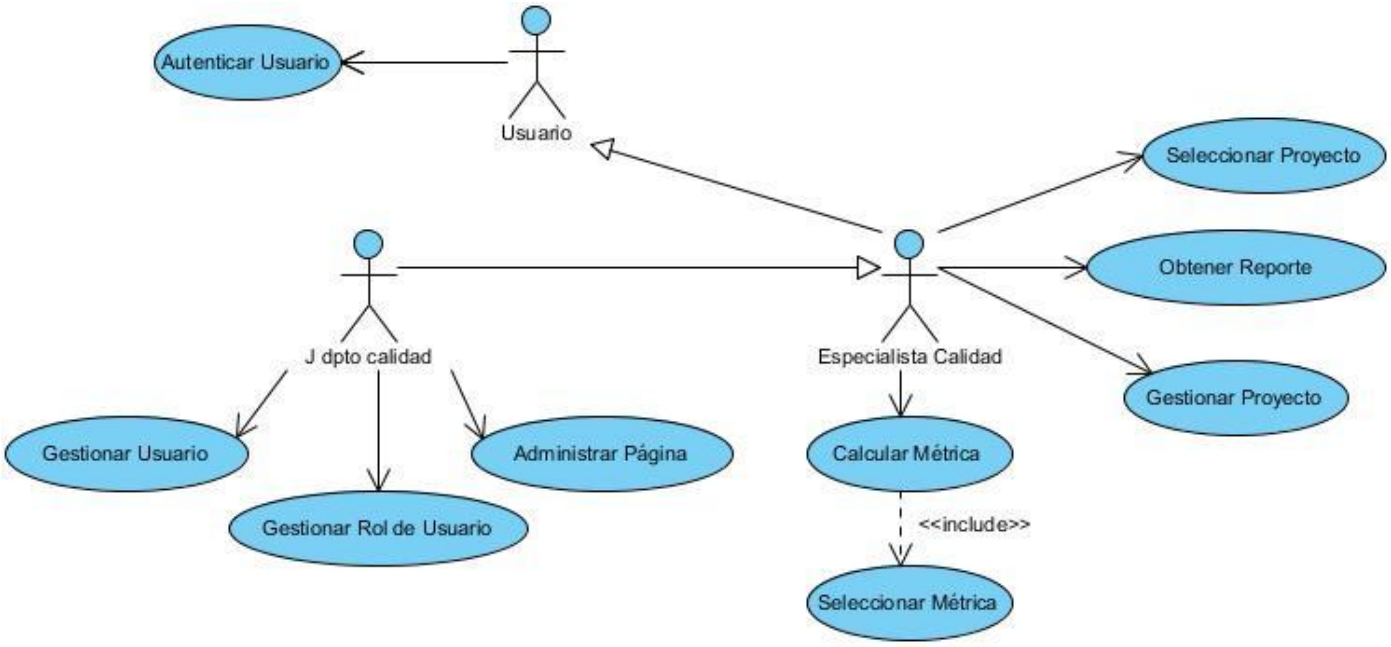


Figura 9 Diagrama de casos de uso del sistema

3.4.3 Especificaciones de los Casos de Uso del Sistema.

Objetivo	Autenticarse en el sistema.

Actores	Usuario (Inicia).	
Resumen	El caso de uso se inicia cuando el usuario decide acceder al sistema y finaliza cuando el sistema le permite al mismo acceder a las páginas según los permisos que tiene asignados.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El usuario debe pertenecer al dominio local.	
Postcondiciones	El usuario se autenticó y pudo acceder al sistema.	
Flujo de eventos		
Flujo básico Autenticar usuario		
	Acción del actor	**Respuesta del sistema**
1.		1.1 El sistema muestra ventana para introducir los datos. (usuario y contraseña).
2.	El actor introduce los datos correspondientes.	2.1 El sistema verifica los datos. 2.2 El sistema muestra interfaz de acuerdo al tipo de usuario (especialista de la calidad o jefe de departamento de calidad). 2.3 Termina el caso de uso.
Flujos alternos		
Nº Evento 1. Datos incorrectos		

	Acción del actor	Respuesta del sistema
		2.2a Emite un mensaje: “No pueden existir campos vacíos” 2.2ª Emite un mensaje: “Usuario y/o clave incorretas”
Requisitos no funcionales	Seguridad	

Tabla 6 Descripción CU Autenticar usuario

Objetivo	**Seleccionar un proyecto para obtener su información.**	
Actores	Especialista de calidad o Jefe de departamento de calidad (Inicia).	
Resumen	El caso de uso se inicia cuando el usuario de calidad desea seleccionar el proyecto a revisar y finaliza cuando lo selecciona.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El usuario de calidad debe estar autenticado.	
Postcondiciones	El usuario seleccionó el proyecto.	
Flujo de eventos		
Flujo básico Buscar proyecto		
	Acción del actor	**Respuesta del sistema**
1.	El actor selecciona la opción “Seleccionar	1.1 El sistema muestra el listado

	proyecto”	de los proyectos existentes.
2.	El actor selecciona el proyecto que desea evaluar.	2.1 El sistema habilita el botón de seleccionar. 2.2 El sistema guarda los datos del proyecto. 2.3 El sistema muestra debajo de los datos del usuario autenticado, el nombre del proyecto seleccionado 2.4 Termina el caso de uso.

Tabla 7 Descripción del CU Seleccionar proyecto

Objetivo	**Calcular las métricas.**
Actores	Especialista de calidad o Jefe de departamento de calidad (Inicia).
Resumen	El caso de uso se inicia cuando el usuario de calidad seleccionó la métrica y finaliza cuando se calcula la métrica.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El usuario de calidad debe haber seleccionado una métrica.
Postcondiciones	El usuario de calidad calculó la métrica.
Flujo de eventos	
Flujo básico Calcular métrica	

	Acción del actor	Respuesta del sistema
1.	El actor introduce los parámetros necesarios. Presiona el botón “Calcular” correspondiente a la métrica.	1.1 El sistema verifica los datos 1.2 El sistema guarda el resultado. 1.3 El sistema actualiza y muestra la tabla de resultados obtenidos por cada métrica calculada a ese proyecto. 1.4 Termina el caso de uso.
Flujos alternos		
Nº Evento 2. Datos incorrectos.		
		1.1ª El sistema muestra mensaje “Solo se pueden introducir números” 1.1ª El sistema muestra mensaje “No pueden existir campos vacíos”

Tabla 8 Descripción del CU Calcular Métrica

Objetivo	**Obtener el reporte de resultados de las métricas.**
Actores	Especialista de calidad o Jefe de departamento de calidad (Inicia).
Resumen	El caso de uso se inicia cuando el usuario de calidad termina de calcular las métricas y desea obtener el reporte de estos resultados.
Complejidad	Alta
Prioridad	Crítico

Precondiciones	El usuario de calidad debe haber calculado las métricas deseadas.	
Postcondiciones	El usuario de calidad obtiene el reporte.	
Flujo de eventos		
Flujo básico Obtener reporte de resultados		
	Acción del actor	**Respuesta del sistema**
1.	El actor selecciona la opción “Obtener Reporte”.	1.1 El sistema obtiene y estructura la información. 1.2 El sistema muestra el reporte. 1.3 Termina el caso de uso.

Tabla 9 Descripción del CU Obtener Reporte

Las restantes descripciones de los casos de uso se encuentran en los anexos.

3.5 Conclusiones del capítulo

En este capítulo se definieron las funcionalidades del sistema, además se especificaron y describieron los casos de usos resultantes, favoreciendo a una mejor comprensión del sistema a desarrollar.

Capítulo 4 Diseño del sistema

En este capítulo se expone el diseño para la solución de la aplicación. Se representan los diagrama de clases diseñadas con sus relaciones, el diagrama entidad relación, la descripción de la arquitectura del sistema y de las tablas de la base de datos.

4.1 Modelo del Diseño

Es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. (16)

4.1.2 Diagramas de clases del diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. (17)

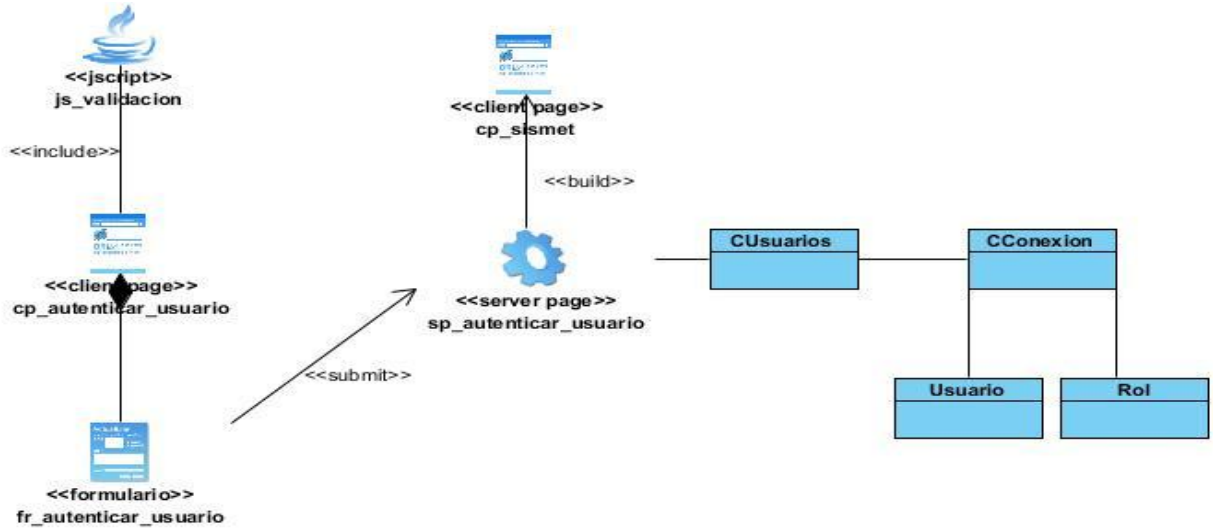


Figura 10 Diagrama de clases del diseño Autenticar usuario

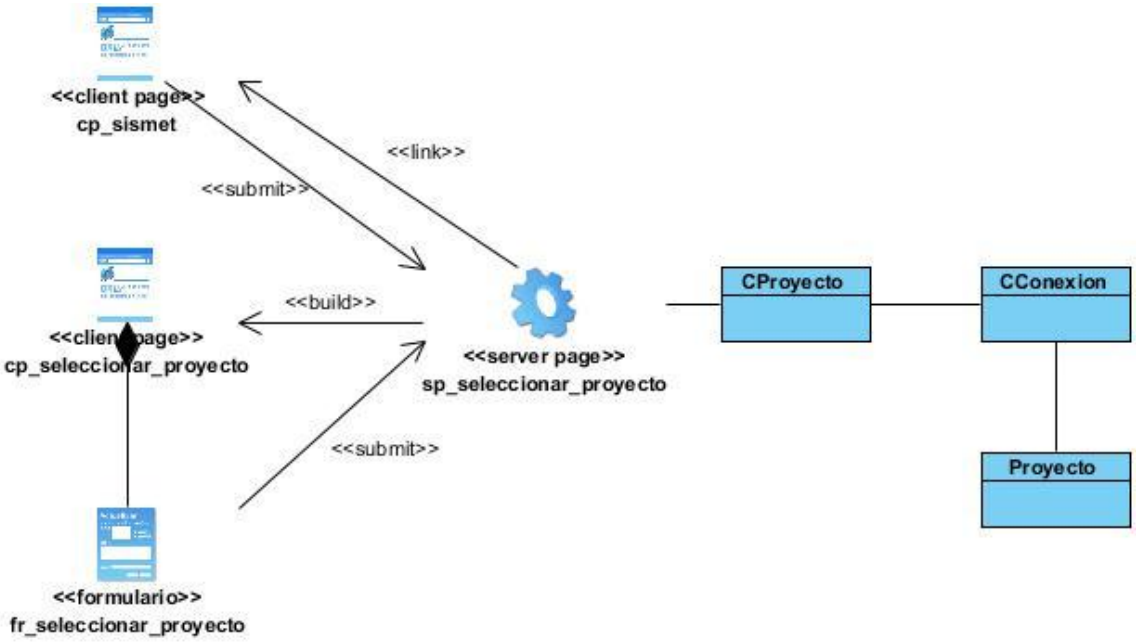


Figura 11 Diagrama de clases del diseño Seleccionar proyecto

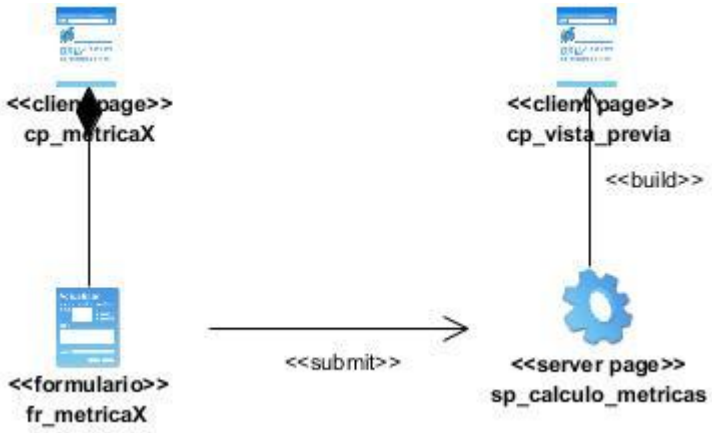


Figura 12 Diagrama de clases del diseño Calcular métricas

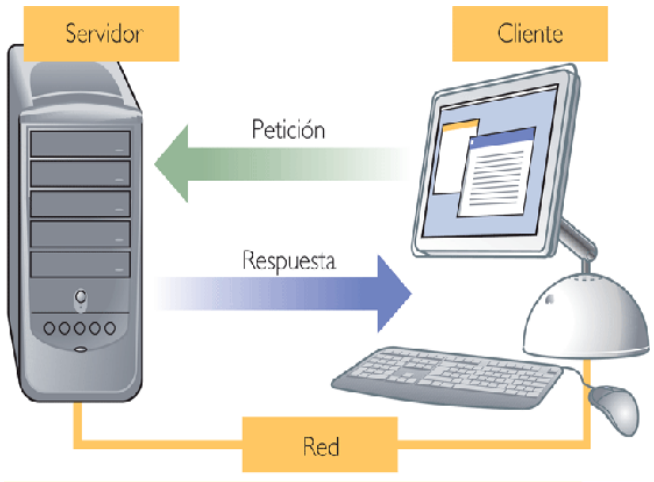
Los restantes diagramas de clases se encuentran en los anexos.

4.2 Arquitectura del sistema

Se denomina arquitectura de software o del sistema a las formas y guías generales que indican la estructura, funcionamiento e interacción entre las partes del software.

4.2.1 Arquitectura Cliente-Servidor

La arquitectura cliente-servidor consiste en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Este tipo de arquitectura a pesar de ser más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras, puede aplicarse también a programas que se ejecutan sobre una sola computadora. Esta arquitectura posee una capacidad de proceso repartida entre el cliente y el servidor, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.



4.2.2 Estilo Arquitectónico Modelo Vista Controlador

El patrón Modelo-Vista-Controlador (MVC) es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios. Este patrón organiza la aplicación en tres modelos separados, el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio, el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información, el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema. (18)

Este estilo arquitectónico presenta varias ventajas:

- ✓ Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado.

- ✓ Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- ✓ La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Se decidió utilizar este estilo para el desarrollo del sistema en cuestión debido a que las características y ventajas anteriormente expuestas permiten que el mantenimiento de las aplicaciones sea más sencillo.
En el sistema en cuestión, la Vista va a estar constituida por el Paquete Presentación, el Controlador por el Paquete Control y el Paquete Clases Entidad y el Modelo por el Paquete Acceso a Datos.

4.3 Diseño de la Base de Datos

Una base de datos sirve para almacenar la información que se utiliza en un sistema de información determinado. Las necesidades y los requisitos de los futuros usuarios del sistema de información se deben tener en cuenta para poder tomar adecuadamente las decisiones anteriores. El diseño de una base de datos consiste en definir la estructura de los datos que debe tener la base de datos de un sistema de información determinado. (19)
Para el diseño de la base de datos se inició por la definición de las clases persistentes, lo que se puede definir como la capacidad de un objeto de mantener su valor en el espacio y tiempo. Una vez realizada la definición de las clases persistentes, se procedió al refinamiento y clasificación de estas clases y sus atributos realizando la construcción del diagrama de clases persistentes.

4.3.1 Modelo lógico de datos

El modelo lógico de datos representa la información que será manejada por el sistema, constituyendo una fuente de información para el modelo físico. Cuando se habla de modelo lógico de datos se hace referencia al diagrama de clases persistentes, donde solo aparecen las clases persistentes a las que hay que especificar detalles estructurales.

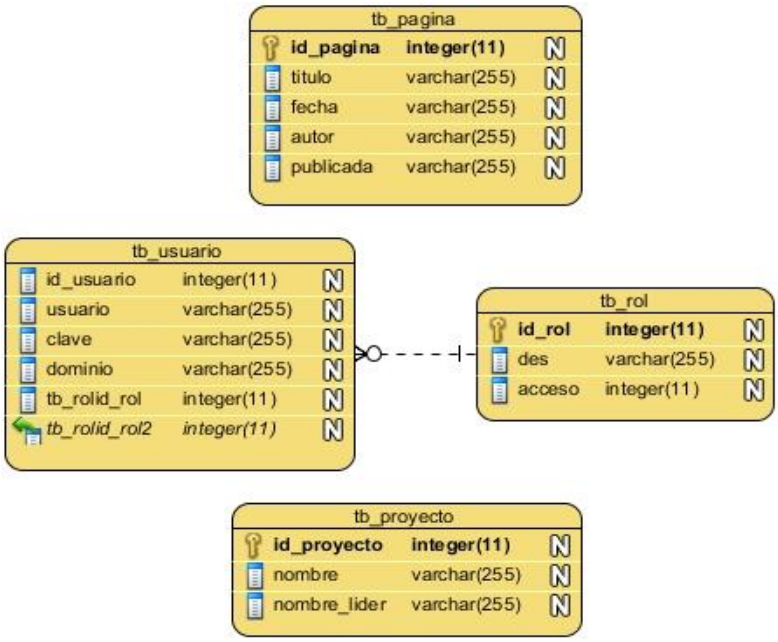


Figura 13 Modelo lógico de datos.

4.3.2 Descripción de las tablas

En este epígrafe se realiza la descripción de las tablas de la base de datos, donde se especifica la función de cada tabla dentro del modelo de datos y las particularidades de sus atributos.

Nombre de la tabla: tb_pagina		
Descripción: Esta tabla es la encargada de almacenar toda la información referente a las páginas.		
Atributo	**Tipo**	**Descripción**
id_pagina *(llave primaria)*	integer (11)	Representa el identificador de la página.
titulo	varchar (255)	Representa el nombre del título de la página.
fecha	varchar (255)	Representa la fecha de publicación de la página.
autor	varchar (255)	Representa el nombre del autor de la página.

publicada	varchar (255)	Representa el estado de la página.

Tabla 10 Descripción de la Tabla página

Nombre de la tabla: tb_usuario		
Descripción: Esta tabla es la encargada de almacenar toda la información referente a los usuarios del sistema.		
Atributo	**Tipo**	**Descripción**
id_usuario *(llave primaria)*	integer (11)	Representa el identificador del usuario.
usuario	varchar (255)	Representa el nombre del usuario.
clave	varchar (255)	Representa la contraseña del usuario.
dominio	varchar (255)	Representa el dominio al cual pertenece el usuario.
tb_rolid_rol *(llave foránea)*	integer (11)	Representa el identificador del rol que tiene asignado ese usuario.

Tabla 11 Descripción de la Tabla usuario

Nombre de la tabla: tb_rol		
Descripción: Esta tabla es la encargada de almacenar toda la información referente a los roles.		
Atributo	**Tipo**	**Descripción**
id_rol *(llave primaria)*	integer (11)	Representa el identificador del rol.
des	varchar (255)	Representa la descripción de ese rol.
acceso	integer (11)	Representa el nivel de acceso que tendrá el rol.

Tabla 12 Descripción de la Tabla rol

Nombre de la tabla: tb_proyecto		
Descripción: Esta tabla es la encargada de almacenar toda la información referente a los proyectos.		
Atributo	**Tipo**	**Descripción**
id_proyecto *(llave primaria)*	integer (11)	Representa el identificador del proyecto.
nombre	varchar (255)	Representa el nombre del proyecto.
nombre_lider	varchar (255)	Representa el nombre del líder del proyecto.

Tabla 13 Descripción de la Tabla proyecto

4.4 Conclusiones del capítulo

El desarrollo de este capítulo posibilitó definir la estructura e interacción del sistema, a partir de la representación de los diagramas de clases diseñadas con sus relaciones, la descripción de la arquitectura del sistema, el modelo lógico de datos y la descripción de las tablas de la base de datos.

Capítulo 5 Implementación y Pruebas

En el presente capítulo se representan los diagramas correspondientes a la implementación del sistema, el diagrama de despliegue y los de componentes. Se representa además para la validación del sistema los casos de pruebas relacionados a los procesos más significativos del sistema desarrollado.

5.1 Implementación

En la implementación se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. Sus principales objetivos consisten en distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue e implementar las clases y subsistemas encontrados en el diseño.

5.1.1 Diagrama de despliegue

El diagrama de despliegue muestra la disposición física de los distintos nodos que componen un sistema y de qué manera se reparten los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

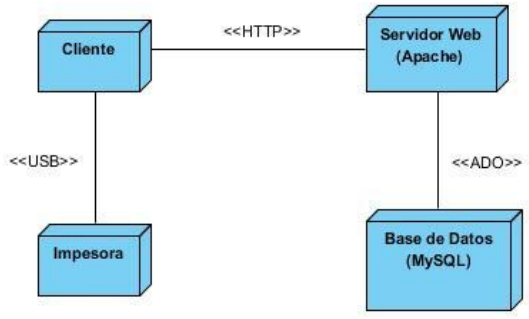


Figura 14 Diagrama de despliegue

5.1.2 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre los componentes del software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de

programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. (20)

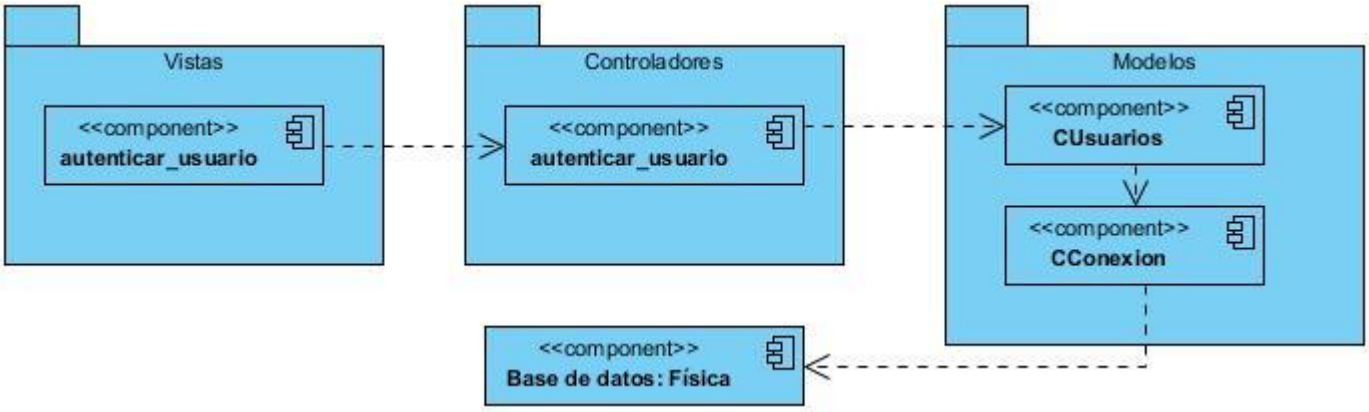


Figura 15 Diagrama de componentes Autenticar usuario

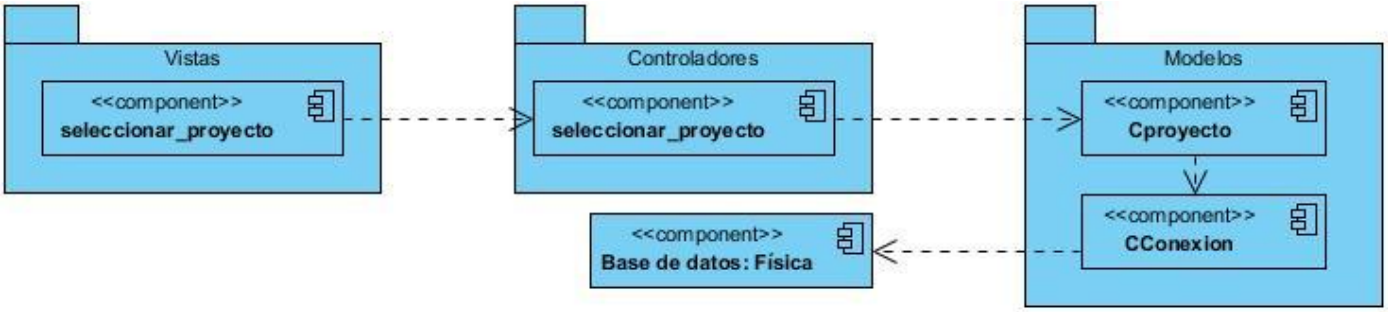


Figura 16 Diagrama de componentes Seleccionar proyecto

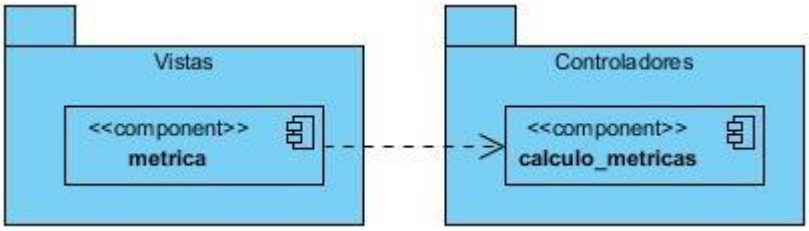


Figura 17 Diagrama de componentes Calcular métricas

Los restantes diagramas de componentes se encuentran en los anexos.

5.2 Pruebas

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados, es por eso que la realización de las mismas a los software es un factor de vital importancia. Antes de liberar el producto es necesario tener la certeza de que este se encuentra libre de errores, aunque se considera que no se puede estar 100% seguro de esto. Probar es un proceso de ejecución de un programa con la intención de descubrir un error. Una prueba tiene éxito si se descubre un error no detectado hasta entonces. (21)

5.2.1 Técnicas de pruebas

Entre la gran variedad de técnicas de pruebas que existen, las más conocidas y empleadas son las pruebas de caja blanca y las pruebas de caja negra.

Prueba de Caja Blanca: Pruebas que se llevan a cabo sobre el código del software.

Entradas: Estándar de programación, archivos fuentes y documentos de diseño.

Salidas: Documentos con defectos encontrados. Estadísticas acerca del seguimiento al estándar, así como de algunas características del sistema, tales como: el tamaño (puntos de función), la complejidad (ciclomática, estructural), la modularidad (cohesión, acoplamiento) y la legibilidad.

Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Prueba de Caja Negra: Pruebas que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada

y se produce un resultado correcto, y que la integridad de la información externa se mantiene. Es completamente indiferente el comportamiento interno y la estructura del programa.

Entradas: Archivos ejecutables, documentos de requerimientos.

Salidas: Documentos con defectos encontrados. Métricas de la aplicación de casos de pruebas.

Estas pruebas permiten detectar:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

5.2.2 Casos de pruebas

Los casos de pruebas son un conjunto de condiciones o variables mediante los cuales se determina si el requisito de una aplicación es parcial o completamente satisfactorio. Un caso de prueba consta de una entrada conocida y una salida esperada, estos son formulados antes de que se ejecute la prueba, donde la entrada conocida debe probar una precondición y la salida esperada debe probar una postcondición.

Los casos de prueba escritos incluyen una descripción de la funcionalidad que se probará, la cual es tomada ya sea de los requisitos o de los casos de uso, un caso de prueba derivado de un caso de uso consiste en una prueba externa al sistema, es decir una prueba de caja negra.

A continuación se muestra la descripción de las secciones por escenarios de los casos de prueba:

Nombre de la sección	Escenarios de la sección	Respuesta del sistema	Resultado 1ra iteración	Resultado 2da iteración
SC1: Autenticar usuario	EC 1.1: Introduce los datos de	El sistema verifica los datos. El sistema muestra interfaz	Satisfactorio	Satisfactorio

	la autenticación. Selecciona la opción de autenticar.	de acuerdo al tipo de usuario (especialista de la calidad o jefe de departamento de calidad).		
	EC1.2: Existen campos incompletos.	No permite la autenticación. Muestra un mensaje de error.	Satisfactorio	Satisfactorio
	EC1.3: Existen campos incorrectos.	No permite la autenticación. Muestra un mensaje de error.	Satisfactorio	Satisfactorio

Tabla 14 Caso de prueba Autenticar Usuario

Nombre de la sección	Escenarios de la sección	Respuesta del sistema	Resultado 1ra iteración	Resultado 2da iteración
SC1: Seleccionar proyecto	EC1.1: Va a la opción de seleccionar proyecto. Selecciona el proyecto que desea.	Muestra debajo del perfil del usuario autenticado el proyecto que se seleccionó.	Satisfactorio	Satisfactorio

Tabla 15 Caso de prueba Seleccionar Proyecto

Nombre de la sección	Escenarios de la sección	Respuesta del sistema	Resultado 1ra iteración	Resultado 2da iteración
SC1: Calcular métricas	EC1.1 Introduce los valores y presiona el botón Calcular.	Verifica los datos. Actualiza y muestra el resultado del cálculo en la tabla de resultados obtenidos por el proyecto para cada métrica.	No satisfactorio Funcionalidad (Para tres de las métricas no mostraba el resultado correcto)	Satisfactorio

	EC1.2 Valores incorrectos.	Muestra un mensaje de error “Solo caracteres numéricos”	Satisfactorio	Satisfactorio

Tabla 16 Caso de prueba Calcular Métricas

Nombre de la sección	Escenarios de la sección	Respuesta del sistema	Resultado 1ra iteración	Resultado 2da iteración
SC1: Obtener reporte	EC1.1 Selecciona la opción de obtener reporte de resultados.	Obtiene y estructura la información. Muestra el reporte con los resultados.	Satisfactorio	Satisfactorio
	EC1.2 No se han calculado.	El sistema muestra mensaje “No hay resultados para mostrar”	Satisfactorio	Satisfactorio

Tabla 17 Caso de prueba Obtener Reporte

Los restantes casos de prueba se encuentran en los anexos.

5.2.3 Resultados de las pruebas

En la primera iteración de las pruebas funcionales realizadas al sistema, se obtuvieron 23 no conformidades, de ellas 9 eran significativas y 11 no significativas. La detección de estas no conformidades, posibilitó la corrección a tiempo de las mismas.

Se llevó a cabo una segunda iteración, con el objetivo de verificar que se le había dado solución a los errores detectados. Teniendo en cuenta que al corregir los errores anteriores, existe la posibilidad de que se generen otros nuevos, es importante verificar en cada iteración con todos los casos de prueba, incluyendo también los que obtuvieron un resultado satisfactorio en todos sus escenarios. Una vez finalizada la 2da iteración de estas pruebas funcionales se confirmó que todas las no conformidades quedaron resueltas.

Los resultados se muestran en la siguiente tabla.

No conformidades	1ra iteración	2da iteración
No significativas	11	0

Formato	3	0
Ortografía	5	0
Errores de interfaz	3	0
Significativas	9	0
Validaciones	4	0
Opciones que no funcionan	5	0

Tabla 18 Resultados de casos de prueba

5.3 Conclusiones del capítulo

En este capítulo se realizó toda la implementación del sistema y se representaron el diagrama de despliegue y los de componentes, propiciando una visión de cómo está distribuido el sistema físicamente. Además se diseñaron casos de pruebas que permitieron comprobar el cumplimiento adecuado de las funcionalidades del sistema.

Conclusiones Generales

Se realizó un estudio detallado de los modelos de calidad que contienen métricas que abarcó los estándares y metodologías que se utilizan en el mundo para el desarrollo de las mismas. Este estudio facilitó la selección del modelo de calidad de la ISO 9126 que propone un grupo de métricas para medir la calidad del producto de software.

Se hizo un análisis de las tecnologías vigentes para desarrollar este tipo de sistema, lo que permitió seleccionar la metodología, plataforma y herramientas a utilizar en el desarrollo del software.

Se modeló el sistema, especificando y describiendo sus casos de uso, lo que permitió obtener todos los artefactos necesarios para la comprensión e implementación del mismo.

Se implementó un sistema que permite el cálculo de las métricas definidas por el modelo de calidad seleccionado, dando solución a la situación problemática existente.

Se realizaron pruebas al sistema implementado, arrojando en la totalidad de los casos resultados satisfactorios, demostrando que cumple con todos los requisitos planteados.

Recomendaciones

Se recomienda

- ✓ La implementación de una nueva funcionalidad que permita al sistema registrar los datos históricos de cada proyecto, de manera que se puedan realizar comparaciones de los resultados obtenidos, permitiendo así indicar como ha sido la evolución o involución de la calidad de cada proyecto.
- ✓ Una vez implementada la funcionalidad anteriormente descrita se le incorpore al sistema la opción de brindar reportes gráficos donde se pueda observar el grado de avance de cada proyecto de forma más perceptible.
- ✓ Implementar una funcionalidad que permita asignar roles a los usuarios UCI, haciendo uso del servicio LDAP.

Trabajos citados

1. Ingenieria de Software VIII. *Ingenieria de Software VIII.* [En línea] 2004. [Citado el: 28 de 04 de 2012.] http://eclases.tripod.com/id18.html.

2. desarrolloweb.com. [En línea] [Citado el: 28 de 04 de 2012.] http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html.

3. **Pereira, Betzabeth, y otros, y otros.** Métricas de Calidad de Software.

4. **Ejiogu, Lem O.** *Five Principles for the Formal Validation of Models of Software Metrics.* 1991.

5. **Rodríguez., Faustino Sánchez.** *Medida del Tamaño Funcional de Aplicaciones Software.* s.l. : Universidad de Castilla-La Mancha, 1999.

6. ISO/IEC . *Guía 2:.* 1996.

7. **Fenton y Pflegger.** *Software Metrics: A rigurous and Practical Approach.* 1997.

8. Official Site of International Organization for Standarization. [En línea] 2009. [Citado el: 02 de 03 de 2012.] www.iso.org.

9. **Rodríguez, Daniel.** *Medición en la ingeniería del software.*

10. [En línea] http://metrics.sourceforge.net/.

11. Campwood Software. [En línea] http://www.campwoodsw.com/.

12. **PHP, Grupo.** EcuRed. [En línea] http://www.ecured.cu/index.php/PHP.

13. **5.0., Manual de referencia de MySQL.** 2006.

14. Lightweight and smart PHP IDE. [En línea] [Citado el: 28 de 05 de 2012.]

15. Entorno Virtual de Aprendizaje.
FASE_DE_INICIO._DISCIPLINA_DE_MODELAMIENTO_DEL_NEGOCIO. [En línea] Universidad de Ciencias Informáticas. [Citado el: 12 de 04 de 2012.] http://eva.uci.cu/.

16. **García Peñalvo, Francisco José, Conde González, Miguel Ángel y Bravo Martín, Sergio.** *Ingeniería del Software Tema 6: Diseño orientado a objetos.* [Power Point] Salamanca España : Universidad de Salamanca, 2008.

17. **Visconti, Marcello y Astudillo, Hernán.** *Fundamentos de Ingeniería de Software.* [PDF]

18. **Gutierrez, Javier J.** *¿Qué es un framework web?* [PDF]

19. **Costa, Dolors Costal.** *Introducción al diseño de bases de datos.* [PDF]

20. **Pressman, Roger S.** *Ingenieria de Software. Un enfoque práctico.* s.l. : Mc Graw Hill, 2001.

21. **Myers, Glen.** *The Art of Software Testing.* 1979. ISBN-10: 0471078786.

22. **Rangel, Karen.** *Programacion Extrema(XP).* [PPT] Venezuela : Instituto Universitario de Tecnologia del estado Bolivar.

23. **Viltres, Wisel Fernández.** Herramienta para la gestión de las métricas de calidad. *Herramienta para la gestión de las métricas de calidad.* Ciudad de La Habana, Cuba : s.n., 2010.

24. **Garzon, Uberney Castañeda.** Buenas Tareas. *Estandar 1061 de 1998.* [En línea] 2010. [Citado el: 11 de 03 de 2012.] www.buenastareas.com/ensayos/Estandar-1068/343079.html.

25. **Cuerda, Xavier García.** Mosaico. *Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto.* [En línea] 29 de 11 de 2004. [Citado el: 12 de 3 de 2012.] http://mosaic.uoc.edu/2004/11/29/introduccion-a-los-sistemas-de-gestion-de-contenidos-cms-de-codigo-abierto/.

26. Sitio oficial PHP. [En línea] [Citado el: 12 de 03 de 2012.] http://www.php.net/.

27. Sitio oficial de Joomla. [En línea] [Citado el: 12 de 03 de 2012.] http://www.joomlaspanish.org.

28. Sitio oficial de Drupal Hispano. [En línea] [Citado el: 12 de 03 de 2012.] http://www.drupal.org.es.

29. **Basili, V R y Rombach., H. D.** *The tame project: Towards improvement-oriented software environments.* 1988.

30. Entorno Virtual de Aprendizaje. *INTRODUCCIÓN A LA DISCIPLINA DE ANÁLISIS Y DISEÑO.* [En línea] Universidad de Ciencias Informáticas. [Citado el: 14 de 04 de 2012.] http://eva.uci.cu/.

31. diseño montes. [En línea] [Citado el: 17 de 04 de 2012.] http://disenomontes.over-blog.es/article-diagramas-de-colaboracion-37559841.html.

32. Tutorial Drupal. *Tutorial Drupal.* [En línea] [Citado el: 15 de 04 de 2012.] http://www.cursosdrupal.com/content/arquitectura.

33. **Garlan, y otros, y otros.** *An introduction to software architecture.* s.l. : Carnegie Mellon University, 1994.

34. Calidad en el desarrollo de software. *Calidad en el desarrollo de software.* [En línea] 2008. http://geeks.ms/blogs/msierra/archive/2008/08/25/_BF00_C_F300_mo-se-mide-la-calidad-en-el-software_3F00_.aspx.

35. **PRESSMAN.** *"Ingeniería de Software. Un enfoque práctico".* Mexico : s.n., 2005.

36. **María A. Mendoza Sanchez.** Metodologías De Desarrollo De Software. [En línea] 7 de Junio de 2004. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

37. **Sosa López, Dailyn .** Casos de Uso del Sistema. [En línea] 2009. http://www.eumed.net/libros/2009c/585/Modelo%20de%20Casos%20de%20Uso%20del%20Sistema.htm.

38. **RUMBAUGH, I.J.G.B.J.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000.

39. Ecured: Requisitos no funcionales. [En línea] 2011. http://www.ecured.cu/index.php/Requisitos_no_funcionales.

40. **Más Rodés , Lic. Raúl .** La Informática en el proceso de enseñanza-aprendizaje de la Física. Alterativa metodológica para su utilización. [En línea] http://www.ilustrados.com/tema/12222/Informatica-proceso-ensenanza-aprendizaje-Fisica-Alterativa.html.

41. **I Delgado, M Hernándeza, M Sánchezb, B Sánchezc, O Chavianod y B Companionie.** UNA APLICACIÓN INFORMÁTICA PARA LA ENSEÑANZA DE LA FÍSICA. [En línea] 2011. http://www.fisica.uh.cu/biblioteca/revcubfi/2011/Vol.28-No.1E/RCF-28-1E-2011-104.pdf.

42. **Gras Martí, Albert , Cano Villalba, Marisa y Soler Selva, Vicent F.** Uso de las NTIC en la enseñanza de la física. [En línea] http://agm.cat/eao/Ntic/msCINE.htm.

43. **Romero, Hugo .** Rational Unified Process. [En línea] 2009. http://www.slideshare.net/genesyss/diseo-de-sistemas.

44. **Rumbaugh, James.** El Lenguaje Unificado de Modelado (UML), Manual de Referencia. [En línea] http://www.intercambiosvirtuales.org/libros-manuales/el-lenguaje-unificado-de-modelado-uml-de-james-rumbaugh-manual-de-referencia.

45. Bizagi BPMN 2.0. [En línea] http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf.

46. Ecured: Herramienta CASE. [En línea] http://www.ecured.cu/index.php/Herramienta_CASE.

47. **Hernández, Prof: Leydis Dayana.** Introducción a las Bases de Datos. [En línea] http://campus.dokeos.com/courses/BDFATLA/document/Introd.alasBasesdedatos.pdf?cidReq=BDFATLA.

48. Investigación sobre PostgresSQL. [En línea] http://es.scribd.com/doc/36570462/postgreSQL-investigacion.

49. Wikipedia: MySQL. [En línea] 2010. http://es.wikipedia.org/wiki/MySQL.

50. **FÉLIX, A. D. C. S.** El servidor de web Apache: Introducción práctica. Apache 1.x y 2.0 alpha. [En línea] 2000. http://acs.barrapunto.org/articulos/trunk/LinuxActual/Apache/apache.pdf.

51. **Valdeli, Ilario.** Elementos de javascript. [En línea] 2006. http://www.htmlpoint.com/javascript/corso/js_02.htm..

52. **Ajona, Guillermo Prado.** Página Web sobre HTML y CSS. [En línea] https://belenus.unirioja.es/~guprado/pagweb/caraccss.html.

53. Procesamiento del lado del servidor. [En línea] 2009. http://prograweb.com.mx/pweb/0301clienteServidor.html.

54. **Sosa López, Dailyn .** [En línea] http://www.eumed.net/libros/2009c/585/Modelo%20de%20Casos%20de%20Uso%20del%20Sistema.htm.

55. **Mamani Poma, Orlando Jimmy.** Desarrollo de un Sistema Informático para la Administración de Expedientes de la Universidad José Carlos Mariátegui. [En línea] http://www.monografias.com/trabajos82/informe-practicas-preprofesionales-ing-sistemas-e-informatica/informe-practicas-preprofesionales-ing-sistemas-e-informatica3.shtml.

56. Software Gaussian-09w. [En línea] http://gaussian-09w.software.informer.com/&usg=ALkJrhhrXljP4AOb-ICNQLNzO1tayWUptw.

57. Ecured: Visual Paradigm. [En línea] 2010. www.ecured.cu/index.php/Visual_Paradigm.

58. Rational Rose Enterprise . [En línea] http://www.rational.com.ar/herramientas/roseenterprise.html.

59. Lenguaje de Programación. [En línea] 2010. http://es.scribd.com/doc/15270823/LENGUAJES-DE-PROGRAMACION-para-el-space.

60. **Reynoso, Billy.** *Profundizando en Estilos de Arquitectura de Software.* 2004.

61. *Material didáctico sobre la resolución de problemas en las asignaturas de Ciencias Exactas.* **Rodríguez Rodríguez, Dr. Luis E., Rodríguez Legrá, Dr. Donaciano y Sierralta Martínez, Prof. Lidia.**

62. Introducción al diseño web. [En línea] 12 de 10 de 2010. [Citado el: 14 de 04 de 2012.] http://programandoenphp.over-blog.com/article-introduccion-al-diseno-web-fundamentos-de-html-58775230.html.

63. **Macias, Julio Cesar Torres.** *Sistema de Tele-Pago.* Ciudad de la Habana : s.n., 2011.

64. **Celis, Ismael.** ESTADOBETA desarrollo web con estándares. [En línea] 2005. [Citado el: 20 de 5 de 2012.]

65. buenas tareas. *Sistemas Gestres de Base de Datos.* [En línea] [Citado el: 12 de 5 de 2012.] http://www.buenastareas.com/ensayos/Sistemas-Gestores-De-Bases-De-Datos/3590500.html.

Anexos

Anexo 1

Las métricas para la medición de la característica Funcionalidad

- **Métrica Idoneidad**

Nombre de la métrica	Lo que se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Adecuación funcional	¿Cuán adecuada es la función evaluada?	Número de funciones idóneas para ejecutar funciones específicas en comparación con el número de funciones evaluadas.	X = 1 - A/B A - Número de problemas detectados en las funcionalidades del sistema. B - Número de especificaciones de requisitos evaluadas.	0 <= X <= 1 A mayor cercanía al 1 resultará más adecuada	Aplicación Especificación de requisitos
Completitud de la implementación funcional	¿Cuán completa ha sido la implementación y su conformidad con la especificación de requisitos?	Ejecutar las pruebas (de caja negra) funcionales de acuerdo con la especificación de requisitos. Cuente el número de funciones perdidas detectadas y compare el resultado con el número de funciones descritas en la especificación de requisitos.	X = 1 - A/B A - Número de funciones perdidas detectadas en la evaluación. B - Número de funciones descritas en especificación de requisitos.	0 <= X <= 1 A mayor cercanía al 1 resultará mejor	Especificación de Requisitos
Cobertura de la implementación funcional	¿Cuán correcta ha sido la implementación	Ejecutar las pruebas funcionales (de caja negra) de acuerdo con la	X = 1 - A/B A - Número de funciones	0 <= X <= 1 A mayor cercanía	Especificación de Requisitos

	funcional?	especificación de requisitos. Cuente el número de funciones incorrectamente implementadas o funciones perdidas detectadas y compare el resultado con el número total de funciones descritas en la especificación de requisitos. Cuente el número de funciones que están completas en relación con las que no lo están.	incorrectamente implementadas o funciones perdidas detectadas. B - Número de funciones descritas en la especificación de requisitos.	al 1 resultará mejor	
Estabilidad en la especificaciones funcionales	¿Cuán estable son las especificaciones funcionales después de entrar a funcionar?	Cuenta el número de funciones que se describen en las especificaciones funcionales que tuvo que ser cambiado después de que el sistema se pone en funcionamiento y compararlo con el número total de funciones que se describen en las especificaciones de requisitos.	X=1-A/B A- Número de funcionalidades actualizadas después de entrar en funcionamiento. B- Número de funcionalidades descritas en la especificación de requisitos.	0<=X<=1 A mayor cercanía al 1 resultará mejor	Especificación de Requisitos

- **Métricas de exactitud**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Exactitud esperada	¿Existen diferencias entre los resultados actuales y los razonablemente esperados?	Ejecutar los casos de pruebas de entrada versus salida y comparar los resultados actuales y los razonablemente esperados. Cuente el número de casos encontrados con diferencias inaceptables en relación con los resultados razonablemente esperados.	X = A/T A - Número de casos encontrados con diferencias entre los resultados razonablemente esperados y aquellos resultantes más allá de lo permisible. T - Tiempo de operación.	0 <= X A mayor cercanía al 0 resultará mejor	Especificación de requisitos Manual de usuario Reporte de las pruebas
Precisión	¿Con qué frecuencia los usuarios finales encontrar resultados con la precisión adecuada?	Registrar el número de resultados con la precisión adecuada.	X=A/T A- Número de resultados encontrados por los usuarios con un nivel de precisión diferente de la necesaria T- Tiempo de Operación.	0<=X A mayor cercanía al 0 resultará mejor	Especificación de requisitos Reporte de las pruebas

- **Métrica de Seguridad**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Prevención de la corrupción de los datos	¿Cuál es la frecuencia de eventos de corrupción de los datos?	Contar las apariciones de eventos de datos mayor corrupción.	a) X=1-A/N A- Número de veces que un evento de corrupción de los datos importantes se produce. N- Número de casos de prueba que provocó los eventos de corrupción de datos.	a) 0<=X<=1 A mayor cercanía al 1 resultará mejor	Reporte de las pruebas Pruebas a las especificaciones de requisitos
Capacidad de control de acceso	¿Cómo controlar el acceso al sistema?	Contar el número de detectar operaciones ilegales en comparación con el número de operaciones ilegales que se encuentran en las condiciones.	X=A/B A- Número de detectar diferentes tipos de operaciones ilegales. B- Número de tipos de operaciones ilegales que están en las condiciones.	0<=X<=1 A mayor cercanía al 1 resultará mejor	Reportes de las pruebas Pruebas especificaciones requisitos

- **Métricas de Funcionalidad**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Cumplimiento funcional	¿Cómo es compatible con la funcionalidad del producto a las normas aplicables, los estándares y regulaciones?	Cuente el número de elementos que requieren de cumplimiento que se han cumplido y comparar con el número de elementos que requieran el cumplimiento en las condiciones especificadas.	X=1-A/B A- Número de elementos cumplidos de las funcionalidades especificadas que no se han aplicado durante las pruebas. B- Número total de elementos cumplidos con funcionalidad especificada.	0<=X<=1 A mayor cercanía al 1 resultará mejor	Descripción del producto Manual de usuario Normas ó estándares que se apliquen Pruebas a la especificación de requisitos y reporte.
Cumplimiento de estándar de interfaz	¿Cómo se cumplen las interfaces con los reglamentos aplicables, las normas y regulaciones?	Cuente el número de interfaces que se cumplieron y la compara con el número de interfaces a exigir el cumplimiento de las especificaciones.	X=A/B A- Número de interfaces con problemas detectados. B- Número total de Casos de pruebas utilizados.	0<=X<=1 A mayor cercanía al 1 resultará mejor	Descripción del producto a cumplir, estándares, normas y regulaciones de las interfaces.

Las métricas para la medición de la característica confiabilidad.

- **Métricas Madurez**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Intensidad de fallos totales contra casos de prueba	¿Cuántos fallos totales fueron detectados durante un período de pruebas definido?	Cuente el número de fallos totales detectados y el número de casos de pruebas.	X = 1-A1 / A2 A1 - Número total de fallos totales detectados. A2 - Número de casos de pruebas ejecutados.	0 <= X En dependencia del estadio de las pruebas. En etapas más avanzadas, mientras más pequeño, mejor.	Reporte de las pruebas
Grado de solución ante fallos totales	¿Cuántas condiciones de fallo total están resueltas?	Cuente el número de fallos totales que no se repitieron en determinado período de pruebas bajo condiciones similares. Mantenga un reporte de solución de problemas describiendo la situación de todos los fallos totales.	X = A1 / A2 A1 - Número de fallos totales solucionados. A2 - Número total de problemas reales detectados.	0 <= X <= 1 A mayor cercanía al 1 resultará mejor, cuantos más fallos totales estén resueltos.	Reporte de las pruebas
Intensidad de fallos	¿Cuántos fallos fueron detectados durante un período de pruebas definido?	Cuente el número de fallos detectados y compute su intensidad.	X = A / B A - Número total de fallos detectados. B - Tamaño del producto.	0 <= X Depende del estadio de las pruebas. En etapas más avanzadas, mientras más pequeño, mejor.	Reporte de las pruebas
Erradicación de fallos	¿Cuántos fallos han sido corregidos?	Cuente el número de fallos resueltos durante el período de pruebas y compárelos con el número total de fallos detectados.	1) X = A1 / A2 A1 - Número de fallos solucionados A2 - Número total de fallos	0 <= X <= 1 A mayor cercanía al 1 resultará mejor (cuanto menos fallos	Reporte de las pruebas

			reales detectados.	queden mejor)	Base de datos
Cobertura de las pruebas	¿Cuántos casos de pruebas requeridos han sido ejecutados detectados durante las pruebas?	Cuente el número de casos de pruebas que han sido ejecutados detectados durante las pruebas y compárelo con el número de casos de pruebas requeridos para obtener una adecuada cobertura de pruebas.	X = A / B A –Número de casos de pruebas que han sido realmente ejecutados, y que representan el escenario de operación durante las pruebas. B – Número de casos de pruebas a ejecutar requeridos para cubrir los requisitos.	0 <= X <= 1 Mientras más cercano al 1, mejor cobertura.	Especificación de requisitos Manual de usuario Reporte de las pruebas
Madurez de las pruebas	¿Está bien probado el producto?	Cuente el número de casos de pruebas que han obtenido un resultado satisfactorio de los casos realmente ejecutados y compárelo con el número total de casos de pruebas requeridos para cubrir los requisitos.	X = 1- A / B A –Cantidad de NC que fueron detectadas por utilizar los casos de pruebas. B – Número de casos de pruebas a ejecutar para cubrir los requisitos.	0 <= X <= 1 A mayor cercanía al 1 resultará mejor	Especificación de requisitos Manual de usuario Reporte de las pruebas

- **Métricas de tolerancia ante fallos**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Evitación de operaciones	¿Cuántas funciones están	Cuente el número de casos de prueba de operaciones	X = A / B	0 <= X <= 1	Reporte de las

incorrectas	implementadas con capacidad de evitación de operaciones incorrectas?	incorrectas que fueron evitadas para que no causaran fallos totales críticos o serios, y compárelo con el número de casos de pruebas ejecutados a los patrones de operaciones incorrectas considerados período de pruebas bajo condiciones similares.	A - Número de ocurrencia de fallos totales críticos o serios evitada. A - Número de casos de pruebas ejecutados a los patrones de operaciones incorrectas (casi causantes de fallos) durante las pruebas.	A mayor cercanía al 1 resultará mejor, cuantas más operaciones incorrectas del usuario sean evitadas.	pruebas
Evitar el fracaso	¿Cómo los patrones de muchas fallas fueron puestos bajo control para evitar fallos críticos y graves?	Cuente el número de patrones de fallos a evitar y compara con el número de patrones de fallas para ser considerado.	X=A/B A- Número de ocurrencias a evitar fallos críticos y graves contra los casos de prueba del patrón de culpa. B- Número de casos de prueba ejecutados del patrón de culpa (casi causando insuficiencia) durante la prueba.	0<=X<=1 A mayor cercanía al 1 resultará mejor, el usuario puede evitar el fracaso más a menudo críticos o graves.	Reporte de las pruebas

- **Métricas de recuperabilidad**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Grado de disponibilidad	¿Cuán disponible está el sistema para su uso durante un periodo de tiempo especificado?	Pruebe el sistema en la producción como ambiente durante un período de tiempo especificado ejecutando todas las operaciones del usuario.	1) X =(To /To+Tr) To – Tiempo de operación. Tr – Tiempo de reparación.	0 <= X <= 1 A mayor cercanía al 1 resultará mejor	Reporte de las pruebas
Tiempo medio de inactividad	¿Cuál es el tiempo promedio en que el sistema se mantiene no disponible cuando ocurre un fallo total y antes de la arrancada gradual?	Mida el tiempo de inactividad cada vez que el sistema se encuentre no disponible durante un período de prueba especificado y compute el tiempo medio.	X = T / N T - Tiempo total de inactividad. N - Número de desastres observados. El peor caso o la distribución del tiempo de inactividad deben ser medidos.	0 < X Cuanto menor sea mejor, el sistema estará inactivo por menos tiempo.	Reporte de las pruebas
Restaurabilidad	¿Cuán capaz es el producto de auto restaurarse luego de un evento anormal o una solicitud?	Cuente el número de de restauraciones exitosas y compárelo con el número de restauraciones probadas requeridas por las especificaciones. Ejemplos de requisitos de restauración son: - función deshacer,	X = A / B A – Número de casos de restauración exitosos. B – Número de casos de restauración probados por los requisitos.	0 <= X <= 1 A mayor cercanía al 1 mejor y el producto es más capaz de de restaurarse en casos definidos.	Reporte de las pruebas

		- función rehacer			

Las métricas para la medición de la característica usabilidad.

- **Métricas de comprensibilidad**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Accesibilidad a demos	¿A qué proporción de demos/tutoriales pueden acceder los usuarios?	Conduzca las pruebas de usuario. Observe el comportamiento del usuario.	X = A / B A - Número de demos/tutoriales a los que pueden acceder los usuarios exitosamente. B - Número total de demos/tutoriales a los que se puede acceder.	0 < X <= 1 A mayor cercanía al 1 mejor	Manuales de usuarios
Comprensibilidad de la función	¿Qué proporción de las funciones de producto el usuario podrá entender correctamente?	Llevar a cabo pruebas de usuario y el usuario entrevista con cuestionarios. Cuente el número de la función de interfaz de usuario cuando el destino sean fácilmente comprensibles para el usuario y compararlo con el número de funciones disponibles para el usuario.	X=A/B A- Número de funciones de la interfaz, cuya finalidad es correctamente descrito por el usuario. B- Número de funciones disponibles en la interfaz.	0<=X<=1 A mayor cercanía al 1 resultará mejor	Manual de usuario

- **Métricas de atracción**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Adaptabilidad de la apariencia de la interfaz	¿Qué proporción de los elementos de la interfaz puede ser, por su apariencia, adaptado por el usuario para la satisfacción del mismo?	Conduzca las pruebas de usabilidad. Observe el comportamiento del usuario.	X = A / B A - Número de elementos de la interfaz del sistema cuya apariencia puede ser adaptada por el usuario. B - Número de elementos de la interfaz del sistema cuya apariencia querría adaptar el usuario.	0 <= X <= 1 A mayor cercanía al 1 resultará mejor	

- **Métricas de Instructibilidad**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Eficacia de la documentación del usuario y / o sistemas de ayuda en el uso	¿Qué proporción de las funciones pueden ser utilizados correctamente después de leer la documentación o el uso de sistemas de ayuda?	Cuente el número de la función que se usa correctamente después de leer la documentación o el uso de sistemas de ayuda y comparar con el número total de funciones	X=A/B A- Número de funciones que se pueden utilizar. B- Total de número de funciones que ofrece.	0<=X<=1 A mayor cercanía al 1 resultará mejor	Manual de usuario Reporte de las pruebas

Las métricas para la medición de la característica Eficiencia.

- **Métricas de Eficiencia**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Cumplimiento de la eficiencia	¿Cómo es compatible la eficacia del producto a las normas, los estándares y regulaciones?	Cuente el número de elemento que requieren de cumplimiento que se han cumplido y comparar con el número de elementos que requieran el cumplimiento según las especificaciones.	X=1-A/B (X=relación de elementos cumplimiento satisfechos en relación a la eficiencia) A- Número de elementos cumplidos de la eficiencia que se especifica que no se han aplicado durante las pruebas. B- Número total de elementos cumplidos de la eficiencia especificado.	0<=X<=1 A mayor cercanía al 1 resultará mejor	Descripción del producto Manual de usuario

- **Métricas de Comportamiento en el tiempo**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Tiempo de respuesta	¿Cuál es el tiempo necesario para completar una tarea específica? ¿Cuánto tiempo se tarda antes de que la respuesta del sistema a una operación especificada?	Iniciar una tarea específica. Medir el tiempo que tarda la muestra para completar su operación. Mantener un registro de cada intento.	T=(Tiempo al obtener el resultado)-(Tiempo al entrar los datos)	0<T Cuanto antes es mejor	Reporte de las pruebas

- **Métricas de rendimiento**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Rendimiento	¿Cómo muchas tareas pueden realizarse con éxito en un período determinado de tiempo?	Calibrar cada tarea de acuerdo a la prioridad dada por objeto. Iniciar las tareas de varios puestos de trabajo. Medir el tiempo que tarda la tarea para completar su operación. Llevar un registro electrónico de cada intento.	X=A/T A- Número de tarea completada. T= Período de tiempo de observación.	0<X El más grande es el mejor	Reporte de las pruebas

Las métricas para la medición de la característica Portabilidad.

- **Métricas de Adaptabilidad**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Capacidad de adaptación al entorno del software	¿Puede el usuario o desarrollador de software adaptarse fácilmente al entorno? El software es lo suficientemente capaz de adaptarse al entorno de funcionamiento.	Observar el comportamiento del usuario o mantenedor cuando el usuario está tratando de adaptar el software al entorno de operación.	X=1-A/B A- Número de funciones operativas de las tareas que no se completa o no dio suficiente para satisfacer el nivel adecuado durante la prueba de funcionamiento combinado el software con el sistema operativo o el software con aplicaciones simultánea.	0<=X<=1 A mayor cercanía al 1 resultará mejor	

			B- Cantidad total de la función que se probaron.		

- **Métricas de Instalabilidad**

Nombre de la métrica	La métrica se propone medir	Método de aplicación	Medición (fórmula)	Interpretación del valor obtenido	Artefacto de entrada en la medición
Facilidad de instalación	¿El usuario puede fácilmente instalar en el entorno de funcionamiento?	Observar el comportamiento del usuario cuando está tratando de instalar el software al entorno de funcional.	X=A/B A- Número de casos que un usuario tuvo éxito en cambiar el funcionamiento de instalación para su conveniencia. B- Número total de casos que un usuario ha intentado cambiar la operación de instalación para su conveniencia.	0<=X<=1 A mayor cercanía al 1 resultará mejor	