Universidad de las Ciencias Informáticas

Facultad 4

Herramienta para la gestión de las métricas de calidad en las pruebas de liberación.

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

Autor: Wisel Fernández Viltres

Tutor (es): Ing. Asnier Enrique Góngora Rodríguez.
Ing. Yoanis Costilla Camejo.

Ciudad de la Habana

2010

Declaración de Autoría

Declaro que soy el único autor del trabajo “Herramienta para la gestión de las métricas de calidad en las pruebas de liberación” y se autoriza al centro CALISOFT de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste se firma el presente trabajo a los ____ días del mes de _____ del año _____.

Autores: _____
Wisel Fernández Viltres.

Tutores: _____
Ing. Asnier E. Góngora Rodríguez.

_____
Ing. Yoanis Costilla Camejo.

Agradecimientos

A mis padres Elba L. Viltres Chávez y Edicto W. Fernández Bauta por su inmenso apoyo, dedicación y por confiar en mí.

A toda mi familia por confiar siempre en mí, y brindarme todo su apoyo.

A todos mis compañeros de estudio por brindarme su apoyo y su amistad, en especial a Leonel Montero Álvarez, Mario M. Falcón García, Yamila Quintana Iglesias, Roberto H. Leiva Ramos, Gerardo Camejo Urtate y Pedro González Serrano.

A nuestros profesores de la UCI que han influido en nuestra preparación como profesional.

A mis tutores los ingenieros Asnier E. Góngora Rodríguez y Yoanis Costilla Camejo por brindarme todo su apoyo, comprensión, y siempre estar ahí conmigo en todo momento.

A la Revolución por crear esta universidad y brindarme la posibilidad de poder estudiar en la misma, abriéndome las puertas hacia el futuro.

A mi tío Julio García Bauta por brindarme su apoyo incondicional todos estos años.

Dedicatoria

A todas las personas que contribuyeron a que este sueño se hiciera realidad, en especial: A mi madre Elba Viltres y mi Edicto Fernández por siempre estar presente cuando los necesito, por brindarme su amor, cariño, dedicación y esfuerzo todos estos años. Por ambos confiar siempre en mí, por ambos ser mi razón de ser y por lo mucho que los quiero.

A mi tía Virginia Blanco, mis abuelas Isabel Bauta y Carmen por brindarme todo su amor, su apoyo y confiar en mí.

A mi prima Eloisa Bauta por brindarme todo su cariño, afecto, apoyo y confianza.

Resumen

La investigación está asociada al Centro de Calidad para Soluciones Informáticas CALISOFT en la universidad de las ciencias informáticas (UCI). En la misma se pretende dar respuesta a la siguiente interrogante: ¿Cómo contribuir a la gestión de las métricas de calidad en las pruebas de liberación? Para el logro de esta investigación se ha implementado una herramienta para automatizar el proceso de gestión de métricas de calidad en las pruebas de liberación diseñando una aplicación web que permita tramitar todos los elementos necesarios para la gestión de las métricas. Esta aplicación le permitirá, al jefe del laboratorio, a los especialistas de calidad, al administrador y al personal autorizado, interactuar a través de la red con la información almacenada en la aplicación, así como hacer reportes de las métricas para llevar un control del estado en que se encuentran los proyectos que están siendo liberados.

Índice.

Índice de Tablas.

Índice de Figuras.

Introducción

En nuestros días las Tecnologías de la Información y la Comunicación (TIC) forman parte de la cultura tecnológica con la que tenemos que convivir, tomando un papel de vital importancia en nuestro desarrollo social. El creciente uso de las TIC, ha dado como resultado gran avance en la industria del software en todo el mundo.

Nuestro país ha visto en las TIC una vía para el desarrollo económico y cultural, tomando como iniciativa llevar un proceso de informatización en la sociedad cubana. Uno de los principales pasos de este proceso fue la creación de los Joven Club de Computación y Electrónica, y la fundación de la Universidad de las Ciencias Informáticas (UCI).

La Industria Cubana del Software, que todavía está en sus inicios, ya es una fuente importante de ingresos para el país. Su principal objetivo es lograr una alta calidad en los productos informáticos para satisfacer las expectativas del cliente.

Según Pressman 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”. Teniendo en cuenta esta definición se puede decir que los requisitos del software son las bases para las medidas de calidad.

La calidad en un marco mundial, es el proceso que permite a las empresas proveer, a sus clientes, los productos y servicios que necesitan para satisfacer sus necesidades, teniendo siempre presente que el cliente es quien define lo que quiere y por tanto debe recibir lo que busca.

La Universidad de las Ciencias Informáticas se encuentra inmersa en el proceso de mejora para lograr la certificación del nivel 2 de CMMI[1]. Para obtener el resultado esperado muchas de las áreas de la universidad se encuentran desarrollando actividades para lograr el mejoramiento. Una de estas áreas es la de calidad, que está referenciada en el Centro de Calidad para Soluciones Informáticas (CALISOFT).

CALISOFT es el centro que brinda servicios informáticos de calidad, garantizando el desarrollo en la producción de software de calidad, siendo el órgano de certificación de software a nivel de

[1] **Capability Maturity Model Integration** es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

universidad.

Entre los servicios que brinda CALISOFT se encuentra la realización de pruebas de software. Este servicio tiene vital importancia, pues con su aplicación permite detectar a tiempo errores en el producto que pueden causar insatisfacción en el cliente. Garantizando que después de la adecuada corrección de los errores el cliente reciba lo que quiere.

A las pruebas de software se le pueden hacer mediciones mediante métricas. La utilización de estas métricas ayuda a mejorar la calidad de los productos. También ayudan a registrar riesgos que se tienen que tener en cuenta en pruebas de software posteriores y a realizar estimaciones más exactas en la planificación de las pruebas.

Una métrica es, según la IEEE "*Standard Glosary of Software Engering Terms*"

"una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado".

En el software hay que regirse por las normas y metodologías, que generalmente, tienen en cuenta las métricas, por la ayuda que brindan en el objetivo de optimizar la calidad del producto. Si no se sigue ninguna metodología, el producto final no tendrá la calidad requerida.

Actualmente en el centro CALISOFT se realizan muy pocas mediciones, que al realizar las pruebas de liberación, permitan comprobar que los artefactos liberados cumplan con las características de calidad establecidas. Las mediciones que existen se realizan de forma manual y trae consigo retraso en las pruebas. Tampoco se han identificado todas las métricas de calidad que deben gestionarse para poder evaluar los artefactos que se liberan, esto incide negativamente en la calidad de las pruebas de liberación y en la satisfacción del cliente. Al no gestionarse las métricas, no se evalúan los atributos de calidad relacionados a ellas, que están establecidos según el modelo de calidad ISO 9126. El cálculo de las métricas actuales se realiza de forma manual y es muy tedioso, además toma mucho tiempo de los especialistas al frente de las pruebas. Por otro lado, los directivos de la universidad no tienen como argumentar de una forma más eficiente, acerca de cuál es el estado en que se encuentran los proyectos que se le han realizado pruebas de liberación, para después tomar decisiones en el desarrollo de los proyectos de la universidad.

La situación antes expuesta lleva al siguiente **Problema a Resolver:** ¿Cómo contribuir a la gestión de las métricas de calidad en las pruebas de liberación?

Para dar respuesta a este problema, se decidió desarrollar una herramienta informática que permita automatizar el proceso de medición en las pruebas de liberación. Por tanto el **objeto de estudio** es el proceso de gestión de las métricas en las pruebas de liberación. Donde se plantea que el **objetivo general** de la investigación es desarrollar una herramienta para la gestión de las métricas en las pruebas de liberación.

El **campo de acción** se centra en una herramienta para la gestión de las métricas de calidad en las pruebas de liberación del Departamento de pruebas de software de CALISOFT.

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

- ✓ Estudio del proceso de gestión de las métricas de calidad en las pruebas de liberación en CALISOFT, atendiendo al estado del arte y las condiciones propias del centro.
- ✓ Estudio de herramientas que gestionen las métricas en las pruebas.
- ✓ Estudio de las métricas de calidad en las pruebas.
- ✓ Entrevistas con los especialistas de calidad del centro CALISOFT.
- ✓ Estudio de los documentos: Plan de pruebas e historial de las pruebas.
- ✓ 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 herramienta para la gestión de las métricas en las pruebas de liberación.
- ✓ 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** de fundamentación teórica contiene los conceptos y fundamentos necesarios

para el dominio del problema y la comprensión del resto del documento.

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

El **tercer capítulo** contiene el diseño del sistema, se define la estructura de la aplicación mediante el modelo de análisis, diagrama de interacción del diseño y diseño de la base de datos.

El **cuarto capítulo** contiene la elaboración del sistema y 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 la gestión 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 la gestión 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

En este capítulo se abordan los principales procesos de medición, sus características y definiciones para entender la propuesta en cuestión. 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. También se analizan las tecnologías y herramientas que se emplean en el sistema propuesto.

1.1. Calidad de software

En la actualidad el desarrollo de las organizaciones depende en gran medida de los sistemas informáticos, es por esto que el desarrollo del software se ha incrementado de forma tan acelerada. Por lo antes expuesto existe la imponente necesidad de construir software de calidad elevada.

Las metodologías enmarcan un conjunto de criterios de desarrollo que sirven de guía en la forma en que se aplica la ingeniería de software. La calidad del software está ligada a los estándares o metodologías, por esto, es importante seguir una metodología.

La Calidad del Software debe implementarse en todo el ciclo de vida del software. Las distintas actividades para la implantación del control de calidad en el desarrollo de software son:

- ✓ Aplicación de metodología y técnicas de desarrollo.
- ✓ Reutilización de procesos de revisión formales.
- ✓ Prueba del software.
- ✓ Ajustes a los estándares de desarrollo.
- ✓ Control de cambios, mediciones y recopilación de información.
- ✓ Gestión de informes sobre el control de calidad.[i]

En todos los ciclos del software está presente la calidad de software, pues es la que indica en que medida el software cumple con los requisitos establecidos. Además la calidad es el punto clave en la satisfacción del cliente.

1.2. Métricas de software.

Las mediciones son fundamentales para cualquier especialidad de ingeniería, y la ingeniería de software no se excluye. Un ingeniero de software puede utilizar las mediciones para evaluar la calidad en los resultados del proyecto y para ayudar a tomar las decisiones más acertadas para contribuir a la mejora del producto.

Aunque los términos medida, medición y métricas se usan a menudo indistintamente, existen diferencias entre ellos. Una medida "proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto" (Pressman 1998). En este concepto Pressman deja ver claramente que es una medida, que se resume en la evaluación de las particularidades de un proceso.

La medición "es el proceso por el cual los números o símbolos son asignados a atributos o entidades en el mundo real tal como son descritos de acuerdo a reglas claramente definidas" (FENTON 1997), también se puede decir que es el acto de determinar una medida.

Las métricas de software proveen información necesaria en la toma de decisiones técnicas. En la figura 1 se ilustra una extensión de esta definición para incluir los servicios relacionados al software como la respuesta a los resultados del cliente.

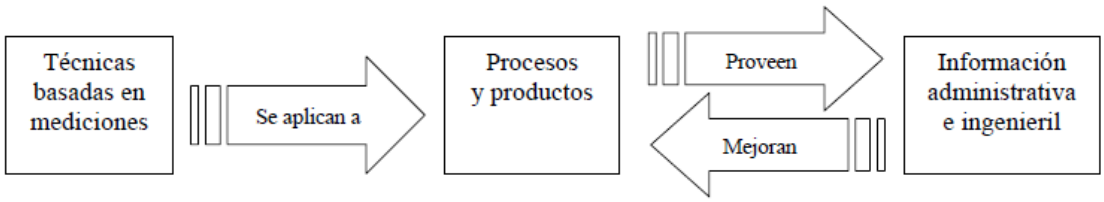


Figura 1: Métricas de software

Según Pressman (1998) las métricas son la maduración de una disciplina que van a ayudar a la (1) evaluación de los modelos de análisis y de diseño, (2) proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y (3) 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.

Las métricas están presentes prácticamente en todos los procesos de desarrollo del software. Son muy útiles en el entorno de desarrollo para la reutilización, pues sirven para controlar la reusabilidad y factores de calidad asociados a este. Es un hecho que los desarrolladores deben poner todo su empeño para recuperar tiempo a través de la reutilización, y las métricas son un medio para evaluar a los desarrolladores en este aspecto, en este caso las métricas de fiabilidad jugarán un papel importante.

1.2.1. Diferentes enfoques de las métricas.

Muchos investigadores han propuesto cientos de métricas para el software, pero no todas proporcionan suficiente soporte práctico para su desarrollo. Algunas necesitan mediciones que son muy complejas, otras son tan esotéricas que pocos profesionales pueden entenderlas, y otras violan las nociones básicas intuitivas de lo que realmente es el software de alta calidad. Es por eso que se han definido una serie de atributos que deben acompañar a las métricas efectivas de software, por lo tanto la métrica obtenida y las medidas que conducen a ello deben cumplir con las siguientes características fundamentales:

- ✓ Simple y fácil de calcular: debería ser relativamente fácil de aprender a obtener la métrica y su cálculo no obligará a un esfuerzo o a una cantidad de tiempo inusual.
- ✓ Empírica e intuitivamente persuasiva: la métrica debería satisfacer las nociones intuitivas del ingeniero de software sobre el atributo del producto en evaluación (por ejemplo: una métrica que mide la cohesión de un módulo debería aumentar su valor a medida que crece el nivel de cohesión).
- ✓ Consistente en el empleo de unidades y tamaños: el cálculo matemático de la métrica debería utilizar medidas que no lleven a extrañas combinaciones de unidades. Por ejemplo, multiplicando el número de personas de un equipo por las variables del lenguaje de programación en el programa, resulta una sospechosa mezcla de unidades que no son intuitivamente concluyentes.

- ✓ Independiente del lenguaje de programación: la métrica debería apoyarse en el modelo de análisis, modelo de diseño o en la propia estructura del programa. No debe depender de la sintaxis o semántica del lenguaje de programación.
- ✓ Un mecanismo eficaz para la realimentación de calidad: la métrica debería suministrar al desarrollador de software información que le lleve a un producto final de superior calidad.
- ✓ Consistentes y objetivas: La métrica debería siempre producir resultados sin ambigüedad. Un tercer equipo debería ser capaz de obtener el mismo valor de métrica usando la misma información del software.

1.2.2. Clasificación de las métricas de software.

Existen muchas formas de clasificar las métricas del software, distintas unas de otras, según los autores. Las métricas pueden ser:

- ✓ Directas o Indirectas.
- ✓ Primarias o Secundarias.
- ✓ Internas o Externas.
- ✓ Públicas o Privadas.
- ✓ Simples o Complejas.
- ✓ De Proceso, de Producto o de Proyecto.
- ✓ Primitivas o Calculadas.

Para esta investigación se escogió la clasificación de las métricas propuestas por Pressman, que distingue 6 categorías o grupos de métricas distintos:

- ✓ 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, y que estudiaremos en detalle en los siguientes apartados.
- ✓ Métricas orientadas a la persona. Aportan información sobre la forma en que la gente desarrolla software.

En la anterior clasificación de las métricas se ve claramente en qué contexto están las métricas de calidad, que forman parte de la investigación.

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

Según la ISO/IEC un estándar es “el documento aprobado por consenso por un organismo reconocido, que proporciona reglas, pautas y/o características para uso común, con el objeto de obtener un óptimo nivel de resultados en un contexto dado.” (ISO/IEC Guía 2: 1996).

Una norma o estándar posee un grupo de especificaciones técnicas determinadas relativas a un área de actividad. Se puede afirmar que es el punto de encuentro entre metodologías, que a través de un proceso de tratamiento, permiten la confrontación de resultados. Los estándares son la guía para que los procesos se realicen siempre con calidad y de forma similar, es decir, las normas o estándares son un modelo, una base, un criterio a seguir. Cada norma tiene un campo de validez que define la aplicación. Por esta razón, muchos productos están sujetos a muchas normas.

1.3.1. Modelos de calidad que contienen métricas.

Entre los principales problemas a los que se deben enfrentar a la hora de hablar de la calidad de un producto de software se encuentra el siguiente: ¿Es realmente posible encontrar un conjunto de propiedades en un producto software que den una indicación de su calidad? Para dar respuesta a esta interrogante aparecen los Modelos de Calidad. En los modelos de calidad, la calidad se define de forma jerárquica. Resuelven la complejidad mediante la descomposición. Los modelos de calidad vienen a ayudar en la puesta en práctica del concepto general de calidad, ofreciendo una definición más operacional.

En el nivel más alto de la jerarquía del modelo de calidad se encuentran los factores de calidad, que representan la calidad desde el punto de vista del usuario. También se les llama Atributos de Calidad Externos. Cada uno de los factores se descompone en un conjunto de criterios de calidad,

estos son atributos que contribuyen al aspecto de la calidad que el factor asociado representa. A estos criterios se les llama Atributos de Calidad Internos.

La ventaja de los modelos de calidad es que la calidad se convierte en algo concreto, que se puede definir, que se puede medir y, sobre todo, que se puede planificar. Los modelos de calidad ayudan también a comprender las relaciones que existen entre diferentes características de un producto de software.[ii]

Modelo ISO 9126.

En 1992, fue propuesto como estándar internacional para medición de calidad de software, el modelo ISO 9126 (*Software Product Evaluation: Quality Characteristics and Guidelines for their Use* es el nombre formal).

El modelo de calidad que describe para la calidad de los productos software está dado al principio, durante la recopilación de requisitos y análisis, por la calidad externa. Ya en la fase de diseño e implementación, la calidad externa se traduce en un diseño teórico, confrontándose con el punto de vista de los desarrolladores sobre la calidad interna. La calidad final (calidad durante el uso) debe ser apropiada para los usuarios y el contexto de uso. Pues no existe una calidad perfecta o absoluta, existe solamente una calidad necesaria y suficiente para un dado contexto.

En ISO 9126 se reconocen seis factores de calidad que se pueden considerar tanto internos como externos:

- ✓ Funcionalidad.
- ✓ Confiabilidad.
- ✓ Eficacia.
- ✓ Usabilidad.
- ✓ Mantenibilidad.
- ✓ Portabilidad.

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

- ✓ Eficacia.

- ✓ Productividad.
- ✓ Seguridad.
- ✓ Satisfacción. [iii]

Se pretende que estas características sean comprensivas, o sea que cualquier componente de la calidad de software se pueda describir como combinación de aspectos de estos factores. A su vez estas características están formadas por sub-características que están refinadas por niveles.

A continuación se dan las definiciones para cada característica de la calidad y las sub-características del software que influyen en aquellas. Para cada característica y sub-característica, la capacidad del software es determinada por un conjunto de atributos internos que pueden medirse.

- ✓ **Funcionalidad:** conjunto de atributos que relacionan la existencia de un conjunto de funciones con sus propiedades especificadas. Las funciones satisfacen necesidades especificadas.
- ✓ **Adecuación:** Atributos que determinan si el conjunto de funciones son apropiadas para tareas especificadas.
- ✓ **Exactitud:** Atributos que determinan que los efectos sean los correctos o los esperados.
- ✓ **Seguridad:** Atributos que miden la habilidad para prevenir accesos no autorizados.
- ✓ **Interoperabilidad:** Atributos que miden la habilidad de interactuar con sistemas especificados.
- ✓ **Cumplimiento:** Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones legales.
- ✓ **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.
- ✓ **Recuperación:** 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.

- ✓ **Cumplimiento:** Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones legales.
- ✓ **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.
- ✓ **Entendimiento:** Atributos que miden el esfuerzo del usuario en reconocer el concepto lógico del software y su aplicabilidad.
- ✓ **Aprendizaje:** Atributos que miden el esfuerzo del usuario en aprender la aplicación.
- ✓ **Operabilidad:** Atributos que miden el esfuerzo de usuario en operar y controlar el sistema.
- ✓ **Atractivo:** Atributos que miden la capacidad del producto de software de ser amigable para el usuario.
- ✓ **Cumplimiento:** Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones legales.
- ✓ **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.
- ✓ **En recursos:** Atributos que miden la cantidad de recursos usados y la duración de tal uso en la ejecución de las funciones.

- ✓ **Cumplimiento:** Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones legales.

- ✓ **Mantenibilidad:** Conjunto de atributos que se relacionan con el esfuerzo en realizar modificaciones.

- ✓ **Analizabilidad:** Atributos que miden el esfuerzo necesario para el diagnostico de deficiencias, para la identificación de las partes que deben ser modificadas.

- ✓ **Facilidad para el cambio:** Atributos que miden el esfuerzo necesario para realizar modificaciones, cambios de fallas en el contexto.

- ✓ **Estabilidad:** Atributos que se relacionan con el riesgo de efectos no esperados en las modificaciones.

- ✓ **Testeabilidad:** Atributos que miden el esfuerzo necesario para validar el software modificado.

- ✓ **Cumplimiento:** Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones legales.

- ✓ **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 provistas para el propósito del software.

- ✓ **Instalabilidad:** Atributos que miden el esfuerzo necesario para instalar el software en el ambiente especificado.

- ✓ **Conformidad:** Atributos que miden si el software se adhiere a estándares o convenciones relacionados con portabilidad.

- ✓ **Reemplazo:** Atributos que se relacionan con la oportunidad y esfuerzo de usar el software en lugar de otro software en su ambiente. [iv]

Para el desarrollo de la investigación se utilizará el modelo ISO 9126, ya que es de los más usados por empresas cubanas para la definición de métricas. Además muchas características de

este modelo hacen referencia a la operación, transición y revisión del software. También es muy flexible y fácil de implementar, pues provee a las empresas un entorno para que lo adecuen a sus necesidades y definan su propio modelo de calidad.

1.3.2. Metodología para el desarrollo de métricas de calidad.

Para definir una métrica primeramente se debe documentar el proceso de desarrollo utilizando la recopilación de datos identificados. Deben establecerse las metas y las métricas para alcanzarlas. Hay que seleccionar las herramientas para el análisis de las métricas, y finalmente, crear una base de datos para archivar la información recolectada, además de las propias métricas.

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

El IEEE 1061-1998 *Standard for a Software Quality Metrics Methodology* provee una metodología para establecer requerimientos de calidad e identificar, implementar, analizar y validar métricas de calidad de productos y procesos software. La metodología es aplicable a todo el software en todas las fases de cualquier estructura de ciclo de vida. En este estándar se establece la mantenibilidad como uno de los factores de la calidad del software. Esta norma está destinada a aquellos asociados con la adquisición, desarrollo, uso, soporte, mantenimiento, y la auditoría de software.

Esta metodología puede ser usada por:

- ✓ Administradores de proyectos para identificar, definir y priorizar los requisitos de calidad de un sistema.
- ✓ Desarrolladores para identificar los rasgos específicos que deben ser tomados en cuenta en el software con el fin de cumplir los requisitos de calidad.
- ✓ Auditores para evaluar si los requisitos de calidad se están cumpliendo.
- ✓ Encargados del soporte del sistema para ayudar en la aplicación de las modificaciones durante la evolución del sistema.
- ✓ Usuarios para ayudar en la especificación de los requisitos de calidad de un sistema.

Específicamente el uso de ésta metodología estándar para medir la calidad permite a la organización:

- ✓ Evaluar el logro de los objetivos de calidad.

- ✓ Establecer los requerimientos de calidad para un sistema en sus inicios.
- ✓ 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.
- ✓ Monitorizar los cambios en la calidad del software si este es modificado.
- ✓ Evaluar la facilidad de cambio en el sistema durante la evolución del producto.

La estructura de las métricas de calidad de software mostrada en la figura 2 está diseñada para ser flexible. Permite adicionar, eliminar y modificar factores de calidad. Cada nivel puede ser ampliado en varios niveles.

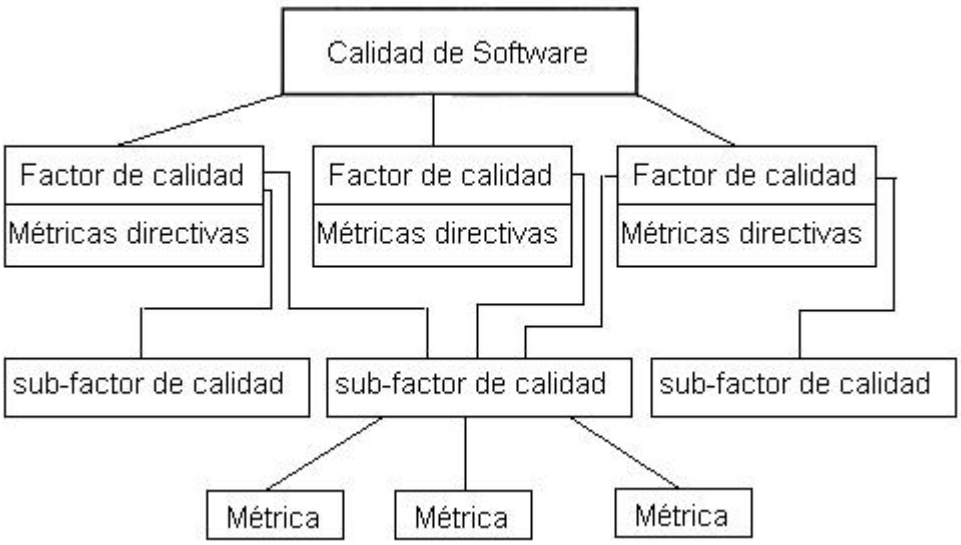


Figura 2: Estructura de las métricas de calidad de software.

Esta metodología es un acercamiento sistemático al establecimiento de requerimientos de calidad e identificación, implementación, análisis y validación de métricas de procesos y productos de software. Comprende 5 pasos:

- ✓ Establecer los requerimientos de calidad de software.
- ✓ Identificar métricas de software.

- ✓ Implementar el sistema de métricas de software.
- ✓ Analizar los resultados de las métricas de software.
- ✓ Validar las métricas de software.

Estos pasos están destinados a ser aplicados iterativamente porque los conocimientos adquiridos a partir de la aplicación de un paso pueden mostrar la necesidad de una nueva evaluación de los resultados de los pasos previos. Cada paso establece las actividades necesarias para completar los resultados indicados.

En la presente investigación se utilizará la IEEE 1061-1998 *Standard for a Software Quality Metrics Methodology* como metodología para la definición de métricas de software por ser una de las más generales, sencillas y explícitas. Además es fácil de implementar y la forma de aplicación que propone involucra un mayor número de personas, ya que no es específica para determinados roles del equipo de desarrollo del proyecto.

1.4. Herramientas para la gestión de métricas.

En el mundo las empresas que brindan servicios de calidad cuentan con herramientas y sitios web que le permiten dar a conocer los productos y servicios que ofrecen, además de brindarlos con mayor calidad.

Entre las compañías líderes en los servicios de calidad de software y pruebas, se encuentra la SQS S.A, empresa que cuenta con un portal web (www.sqs.es). Esta empresa ha alcanzado su objetivo a través de servicios de consultoría, programas de formación y el desarrollo de nuevas herramientas. SQS enfoca sus servicios de consultoría de calidad de software en mejorar y asegurar la calidad de los desarrollos de sus clientes. Entre los servicios que brindan se encuentran los siguientes:

- ✓ Evaluación.
- ✓ Optimización de procesos.
- ✓ Validación y verificación.
- ✓ Laboratorio de pruebas.

SQS desarrolla herramientas para la automatización de procesos de validación y la gestión de

requisitos, como:

- ✓ SQS TestWORKFLOW.
- ✓ SQS AgileREQ.
- ✓ SQS Q-Val.
- ✓ SQS interCENELEC.[v]

Por otra parte IBM, empresa especializada en el desarrollo de herramientas software, desarrolló el *Rational Quality Manager Express Edition*. Herramienta que se destaca en la gestión de pruebas y gestión de la calidad para ofrecer software orientado a la calidad.

Rational Quality Manager Express Edition se diseñó para la colaboración entre pequeñas y medianas empresas, ya que les permite compartir información, usar la automatización para agilizar las programaciones de proyectos y generar informes de mediciones sobre los proyectos con el fin de tomar decisiones bien fundamentadas. Además contribuye a que tengan un mejor control de sus proyectos mediante la provisión de métricas confiables y oportunas. También soporta gran cantidad de funciones de usuario como: gerente de pruebas, arquitecto de pruebas, líder de pruebas y gerente de laboratorio.

Por las características de generación de informes incluidas en *Rational Quality Manager*, se puede personalizar los contenidos y presentar información en paneles de control personalizables. Cada usuario puede personalizar los contenidos y la disposición de la página de inicio y, por lo tanto, potenciar la reunión automatizada de datos con extensiva información filtrada. Esto resulta en un estado de equipo que se actualiza constantemente y todos pueden ver.

El Rational Quality Manager es utilizado fundamentalmente para:

- ✓ Pruebas basadas en los riesgos para dar prioridad a las funciones y a las características para que se analicen en función de la probabilidad o impacto de los errores y, al mismo tiempo, dar soporte a las mejores prácticas de la gestión de riegos.
- ✓ Informe métrico ROI integrado para laboratorios de pruebas y activos de pruebas conforme al MCIF (marco de mejora de capacidad medido).
- ✓ Plan de pruebas del ciclo de vida que define los roles, procesos y propiedad entregable, y automatiza el flujo de trabajo y activos.
- ✓ Ofrece soporte a equipos pequeños y medianos de hasta 5 miembros.

- ✓ Recopilación de datos automatizada mediante generación de informes con una amplia aplicación de filtros.[vi]

1.5. Tecnologías informáticas y metodologías de desarrollo.

1.5.1. Portales web.

Portal es un término usado para referirse al sitio web como un sitio principal para personas que se conectan al *World Wide Web*. Un sitio web no alcanza el rango de portal solo por ser un sitio robusto. Un portal web es una especie de plataforma para la navegación web, los portales pueden ser horizontales o verticales.

Los sitios definidos como portales horizontales se basan en la información universal y amplia que se ofrece para un público general, aunque puede tener secciones para usuarios especializados. El objetivo principal es informativo, puede ser el caso de los sitios de prensa, subastas y otros.

Los sitios de contenido vertical basan su funcionamiento en contenidos especializados dirigidos a determinado público, en este caso la información es de utilidad para personas que cumplan con un perfil, es el caso de los sitios de especialidades académicas.[vii]

Los portales pueden entonces llegar a ser muy especializados, como es el caso de los portales verticales o bien muy genéricos y amplios como los portales horizontales.

En la presente investigación se desarrollará un portal vertical, debido a que se caracterizará por brindar información sobre las métricas en las pruebas de liberación del laboratorio industrial de pruebas de software de CALISOFT.

1.5.2. Sistema de Gestión de Contenido (CMS).

El contenido es la información que se encuentra en la web. La web que alcanza mayor éxito es la que ofrece contenidos interesantes y los actualiza periódicamente. Ahora, ¿Qué es un Sistema de Gestión de Contenido (*Content Management System*, abreviado CMS)? Es un framework que permite crear estructuras de soporte utilizado en la gestión de contenidos, principalmente páginas web.

1.5.2.1. Joomla

Joomla es un sistema de gestión de contenidos, entre sus principales virtudes está permitir editar el contenido de un sitio web de forma sencilla. Es una aplicación de código abierto programada en

PHP bajo la licencia GPL. Joomla trabaja tanto en Internet como en intranets y requiere una base de datos MySQL, así como un servidor, Apache preferiblemente.

Joomla surge como el resultado de una mejora de Mambo, de la corporación Miro de Australia, quien mantenía la marca del nombre Mambo en esa época y el grupo principal de desarrolladores. Joomla nace con esta división el 17 de agosto de 2005.

Entre las características de Joomla se destacan las siguientes:

Componentes: cada página del sitio está formada por elementos de navegación, elementos accesorios, información variada, y un contenido principal gestionado por un componente. Los componentes son el nivel intermedio entre el núcleo y el usuario. Los componentes son los que manipulan contenidos para presentarlos de una u otra forma.

Menús e ítems de menús: los menús son contenedores de los ítems de menú, que son los encargados de enviar al usuario a una vista de un componente. Los menús pueden anidarse.

Módulos: los módulos son elementos para mostrar en la página otros contenidos además del componente, cada elemento presentado es una “instancia” de un tipo de módulo. Se puede definir módulo como “pequeño paquete de información” que se ubica en lugares definidos por la plantilla. Hay distintos tipos de módulo y cada uno de ellos puede configurarse para que muestre contenidos diferentes.

Plantillas: las plantillas establecen la relación entre qué se muestra y cómo se muestra. Por tanto las plantillas son las responsables de la “estética” del sitio. Las plantillas pueden variar dentro del sitio web, asociándolas a ítems de menú. Al haber plantillas muy distintas, puede suceder que dentro del sitio la apariencia cambie mucho.

1.5.2.2. Drupal.

Drupal es un programa libre, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema.

El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar diferentes tipos de sitio web.

Drupal posee una serie de características que lo distinguen de los demás CMS, las cuales se describen a continuación.

Hooks: La documentación de Drupal tiene una sección dedicada a los *Hooks*. Los *Hooks* son el mecanismo que provee Drupal para interactuar con los distintos procesos que se ejecutan en un sitio web. Conocer su funcionamiento es fundamental para cualquier programador de módulos así como también para aquellos diseñadores o *Themers* que deseen modificar aspectos al parecer imposibles de lograr.

Módulos: La comunidad de Drupal ha contribuido con infinidad de módulos que proporcionan funcionalidades como página de categorías, autenticación mediante chat, mensajes privados y bookmarks, entre otros.

Personalización: Un robusto entorno de personalización está implementado en el núcleo de Drupal. Tanto el contenido como la presentación pueden ser individualizados de acuerdo con las preferencias definidas por el usuario.

Gestión y autenticación de usuarios: Los usuarios se pueden registrar e iniciar sesión de forma local o utilizando un sistema de autenticación externo como Jabber, Blogger, LiveJournal u otro sitio Drupal. Para su uso en una intranet, Drupal se puede integrar con un servidor LDAP.

Permisos basados en roles: Los administradores de Drupal no tienen que establecer permisos para cada usuario. En lugar de eso, pueden asignar permisos a un rol y agrupar los usuarios por roles.

Objetos de Contenido (Nodos): El contenido creado en Drupal es, funcionalmente, un objeto (Nodo). Esto permite un tratamiento uniforme de la información, como una misma cola de moderación para envíos de diferentes tipos, promocionar cualquiera de estos objetos a la página principal o permitir comentarios o no sobre cada objeto.

Plataforma Independiente de la base de datos: Aunque la mayor parte de las instalaciones de Drupal utilizan MySQL, existen otras opciones. Drupal incorpora una capa de abstracción de base de datos que actualmente está implementada y mantenida para MySQL y PostgresSQL, aunque permite incorporar fácilmente soporte para otras bases de datos.

Multiplataforma: Drupal ha sido diseñado desde el principio para ser multi-plataforma. Puede funcionar con Apache o Microsoft IIS como servidor web y en sistemas como Linux, BSD, Solaris,

Windows y Mac OS X. Por otro lado, al estar implementado en PHP, es totalmente portable.

Sistema de Cache: El mecanismo de cache elimina consultas a la base de datos incrementando el rendimiento y reduciendo la carga del servidor.

Se utiliza Drupal para desarrollar la propuesta de solución del laboratorio tecnológico de pruebas de software de CALISOFT. Por las características de Drupal, como son la flexibilidad, taxonomía, rendimiento y escalabilidad. También es un CMS desarrollado en el lenguaje de programación PHP, lenguaje que se usa para el desarrollo de la herramienta, y del cual se abordará más adelante. Además Drupal es multiplataforma y libre.

1.5.3. Lenguaje de programación PHP.

PHP (*Hypertext Pre-processor*) es un lenguaje de programación interpretado del lado del servidor, diseñado originalmente para la creación de páginas web dinámicas. Puede ser utilizado en casi todos los sistemas operativos y plataformas de forma gratuita.

PHP es usado especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida.

Ventajas.

- ✓ Es un lenguaje multiplataforma.
- ✓ Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- ✓ El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- ✓ Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- ✓ 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.

- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Permite aplicar técnicas de programación orientada a objetos.
- ✓ No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- ✓ Tiene manejo de excepciones (desde PHP5).

Desventajas.

- ✓ Como es un lenguaje que se interpreta en ejecución para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser ocultado. La ofuscación es una técnica que puede dificultar la lectura del código pero no la impide y aparte en ciertos casos representa un costo en tiempos de ejecución.

Se utiliza como el lenguaje de programación PHP por su facilidad de conexión con bases de datos, además de poder ser utilizado junto con Drupal.

1.5.4. Metodologías.

La metodología, del griego metá “más allá”, odós “camino” y logos “estudio”, hace referencia al conjunto de procedimientos basados en principios lógicos, utilizados para alcanzar una gama de objetivos que rigen en una investigación científica o en una exposición doctrinal.[viii]

Una metodología debe ser adaptable como para aplicarse a diferentes proyectos, y bastante sencilla para que no sea muy difícil de utilizar. Pero al mismo tiempo debe ser lo suficientemente completa para que su utilización por parte de un equipo sea de provecho.

Proceso Unificado Racional (RUP).

El Proceso Unificado Racional (*Rational Unified Process* en inglés, abreviado RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

“El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML) para preparar todos los esquemas de un sistema software. De hecho, UML, es una parte

esencial del Proceso Unificado – sus desarrollos fueron paralelos". (G.Booch, Rumbaugh, & Jacobson, 1999)

RUP se caracteriza por:

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

RUP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada iteración, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante.

Fase de inicio: se logra un acuerdo entre los interesados teniendo en cuenta el ciclo de vida para el proyecto generando el cuerpo del proyecto:

Casos de negocios.

Síntesis de arquitectura posible.

Define el alcance del proyecto.

Fase de elaboración: se establece la estructura base para la arquitectura del sistema, proporciona el diseño del mismo y el desarrollo de la siguiente fase. En esta fase se confecciona:

Plan del proyecto.

Especificación de características.

Arquitectura base.

Fase de construcción: se construye el producto, completando el desarrollo del sistema basado en la estructura base de la arquitectura.

Fase de transición: transición del producto a la comunidad del usuario, en si garantiza que el software esté listo para entregar al usuario.

RUP define nueve disciplinas a realizar en cada fase del proyecto:

- ✓ Modelado del negocio.
- ✓ Análisis de requisitos.
- ✓ Análisis y diseño.
- ✓ Implementación.
- ✓ Prueba.
- ✓ Distribución.
- ✓ Gestión de configuración y cambios.
- ✓ Gestión del proyecto.
- ✓ Gestión del entorno.

RUP ejecuta un ciclo de vida iterativo incremental, donde en cada iteración se produce un producto ejecutable.

Programación Extrema (PX).

La programación extrema o *eXtreme Programming* (XP), es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

Las características fundamentales del método son:

- ✓ Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- ✓ Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.

- ✓ Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que hay mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- ✓ Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- ✓ Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- ✓ Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- ✓ Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- ✓ Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

Se usará la metodología RUP para el proceso de desarrollo del sistema de gestión de las métricas en las pruebas de liberación en el laboratorio tecnológico de pruebas de software de CALISOFT. Debido a que es la metodología más utilizada en la universidad y permite mejor entendimiento del negocio del sistema a desarrollar.

1.5.5. Herramientas.

1.5.5.1. Visual Paradigm.

Visual Paradigm es una herramienta para el modelado UML que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Esta herramienta de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo.

Visual Paradigm permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos de UML.

Características de Visual Paradigm:

- ✓ 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 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 IDEs.
- ✓ Disponibilidad en múltiples plataformas.

Se utiliza Visual Paradigm para el modelado UML, pues es la herramienta CASE utilizada en la universidad para modelar y documentar los productos de software.

1.5.5.2. PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. El desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

Características de PostgreSQL:

- ✓ Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas, común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.
- ✓ Amplia variedad de tipos nativos: PostgreSQL provee nativamente soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas, direcciones IP (IPv4 e Ipv6), bloques de direcciones estilo CIDR, direcciones MAC y arrays.
- ✓ Disparadores (*triggers*): Un disparador o *trigger* se define como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.
- ✓ Integridad transaccional.
- ✓ Herencia de tablas.
- ✓ Soporta el uso de índices, vistas y procedimientos almacenados en múltiples lenguajes.
- ✓ Es multiplataforma, funciona en todos los sistemas operativos importantes, incluyendo Linux, UNIX y Windows.

Se utiliza PosgreSQL como gestor de base de datos por ser robusto y libre, además implementa el estándar SQL92/SQL99.

1.5.5.3. Servidor Apache

Servidor Apache es un servidor web HTTP de código abierto para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP y la noción de sitio virtual.

La arquitectura del servidor Apache es muy modular. El servidor 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: Módulo con 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.

Las principales ventajas que nos brinda este servidor son:

- ✓ Corre en una gran multitud de sistemas operativos por lo que se considera multiplataforma lo que lo hace prácticamente universal.
- ✓ Es una tecnología gratuita de código abierto.
- ✓ Es altamente configurable y de diseño modular. Es sencillo ampliar las capacidades de este servidor.
- ✓ Trabaja con Perl, PHP y otros lenguajes de script.
- ✓ Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.

Se utilizará Apache como servidor de base de datos por ser multiplataforma y gratuito, además trabaja con PHP que es el lenguaje que se utilizará en el desarrollo de la solución propuesta en la investigación.

1.5.5.4. NetBeans

NetBeans IDE es un producto libre y gratuito sin restricciones de uso. La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

Las características del IDE que integra el desarrollo en el lenguaje PHP (*NetBeans IDE Early Access for PHP build*) son:

- ✓ Soporta desarrollo web local.
- ✓ Soporta la ejecución de scripts PHP en línea de comandos.

- ✓ Creación y personalización de proyectos.
- ✓ Reconocimiento por defecto de la ubicación raíz de documentos en diferentes sistemas operativos.
- ✓ Ejecuta y realiza la acción de depuración del proyecto.
- ✓ Asistente para la creación de archivos PHP nuevos.

Se utiliza NetBeans IDE como plataforma para el desarrollo del lenguaje de programación web PHP por ser libre y robusto.

1.6. Conclusiones del capítulo.

- ✓ La medición en las pruebas de liberación es la base para detectar las desviaciones del rendimiento aceptable en los procesos y producto de software, brinda la posibilidades de identificar y priorizar las principales preocupaciones, y mejorar la calidad del producto.

- ✓ En la propuesta que se presenta a continuación, se utiliza como metodología para definir e implementar las métricas seleccionadas la IEEE 1061-1998 Standard for a Software Quality Metrics Methodology y como modelo de calidad el ISO/IEC 9126-1 del 2005 Parte 1: Modelo de Calidad, dado que los mismos facilitan el proceso de prueba de un software.

- ✓ Se utiliza como metodología de desarrollo RUP, ya que da una mejor definición de los procesos para el desarrollo de software, sin importar el tamaño que este posea.

- ✓ Se utiliza la herramienta case Visual Paradigm para llevar a cabo el modelado de los procesos, brindando soporte para la utilización de códigos PHP.

- ✓ Se utilizan para el desarrollo e implementación de la solución propuesta herramientas como: NetBeans, Servidor Apache y PostgreSQL.

- ✓ La herramienta se desarrolla sobre el CMS Drupal.

Capítulo 2: Características del sistema

En este capítulo se dará una breve descripción del rol que desarrolla el proceso de gestión de las métricas, así como el papel que juegan otras personas que puedan estar involucradas. Se definirá además la información manejada y la propuesta del sistema. Se identificarán los requisitos funcionales y no funcionales, los casos de uso con sus descripciones, los diagramas de actividades y el modelo de objeto.

2.1. Modelo del negocio.

Se utilizara cómo flujo de trabajo el Modelamiento del Negocio, proceso que permite obtener una visión de la organización que permita definir los procesos, roles y responsabilidades de la organización. También define los modelos para el desarrollo de un software, así como comprender la estructura y la dinámica de la organización en la cual se va a implementar el 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, además de derivar los requerimientos del sistema que va a soportar la organización. Además establece las competencias que se requieren de cada proceso: sus trabajadores, sus responsabilidades y las operaciones que llevan a cabo.

La descripción del negocio propuesto tiene entre sus actividades principales la identificación de los procesos del negocio, así como la redacción y especificación de los casos de uso, la identificación de los roles y entidades que ejecutan las realizaciones de los casos de uso del negocio y detallar la definición de las entidades y las responsabilidades de los roles del negocio.

Se selecciona el modelo del negocio puesto que es más amplio que el modelo de dominio, y es necesario hacer un modelo completo del negocio para entender mejor lo que se pretende realizar. Además se especifican las responsabilidades de las personas que ejecutan las actividades, posibilidad que no brinda el modelo del dominio.

2.1.1. 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. Además todo lo que interacciona con el ambiente del negocio se modela con actores. Cada actor humano expresa un rol, no una persona específica. El mismo modela algo fuera del negocio, se involucra con un caso de uso, al menos como regla, también tiene una descripción y un nombre que explica su rol en relación al negocio.

Tabla 1: Actores del negocio.

Actor	Descripción
Directivos de la universidad	Le solicitan al jefe del laboratorio de pruebas, el estado de los proyectos que están siendo revisados hasta ese momento. Ellos pueden ser los decanos de cualquier facultad, vicedecanos, el rector, vicerrector, etc.
Solicitante	Asesor de calidad de la facultad, vicedecanos de producción o toda aquella persona que solicite la revisión del software.

2.1.2. Trabajadores del negocio.

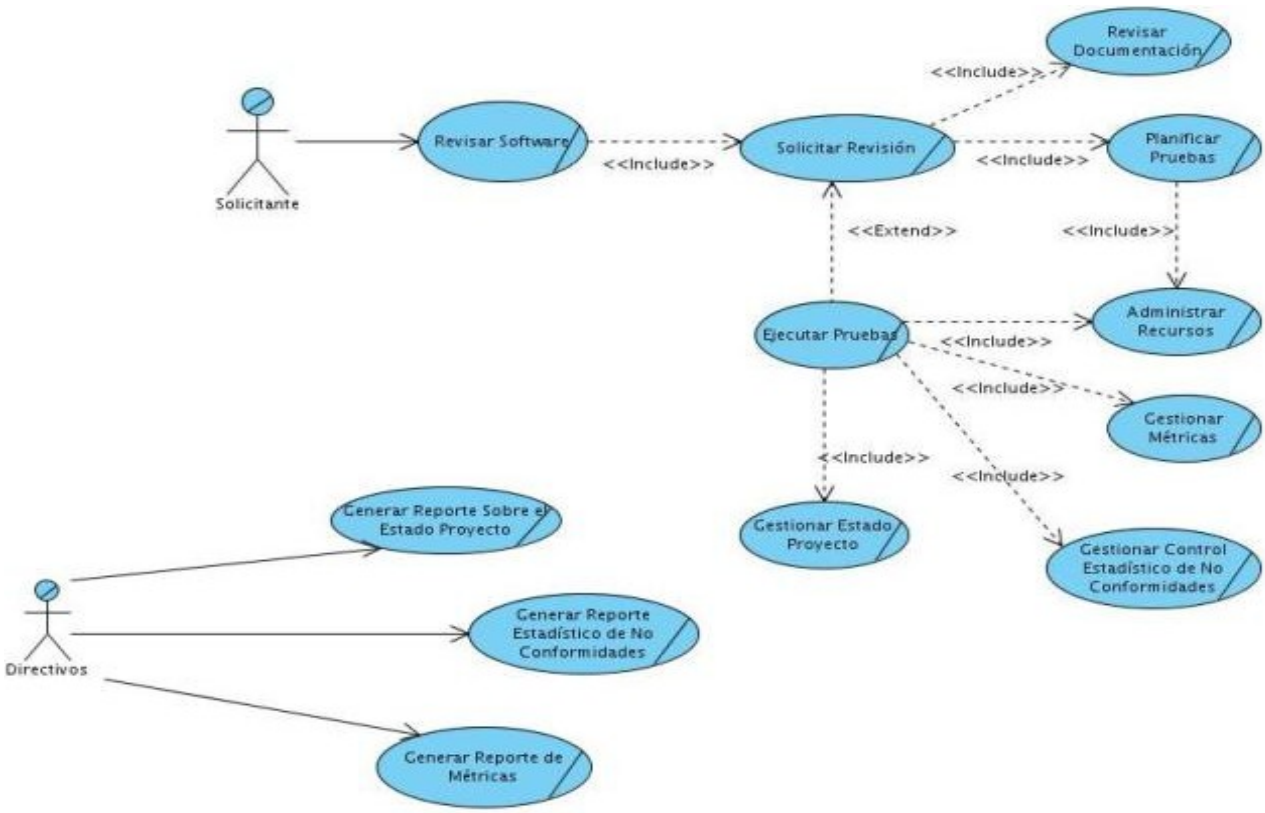
Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

Tabla 2: Trabajadores del negocio.

Trabajador	Descripción
Jefe del Laboratorio	Encargado de elaborar un reporte general sobre el estado de todos los proyectos que se están revisando, para luego enviárselo a los directivos de la universidad. Revisa los planes de mediciones, controla los resultados de las métricas así como la

	evaluación final del software.
Especialista de Calidad	Encargado de recoger en un reporte el estado de los proyectos que estén revisando en ese momento, para luego enviárselo al jefe del laboratorio. Encargado de calcular las métricas, aplicarlas y obtener los resultados necesarios para los reportes.

2.1.3. Diagrama de Casos de Uso del negocio.



2.1.4. Identificar los Casos de Uso del negocio más significativos.

- ✓ Gestionar Métricas.
- ✓ Generar Reporte de Métricas.

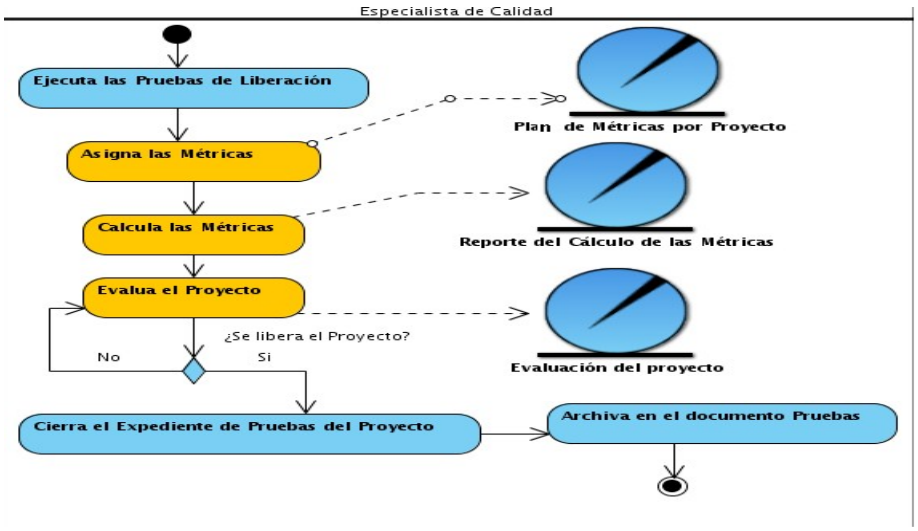
2.1.5. Descripción del negocio: CU1 Gestionar Métricas.

Durante el proceso de pruebas, cada vez que se termina una iteración, los especialistas de calidad al frente de las pruebas de cada proyecto deben de recoger los datos por cada artefacto que estén liberando, así como calcular las métricas que fueron asignadas al proyecto. Esto se almacena en el expediente de las pruebas de liberación de cada proyecto para después realizar los diferentes reportes que se soliciten por parte de los directivos de la universidad o para mejorar el proceso de pruebas de liberación para los proyectos a revisar.

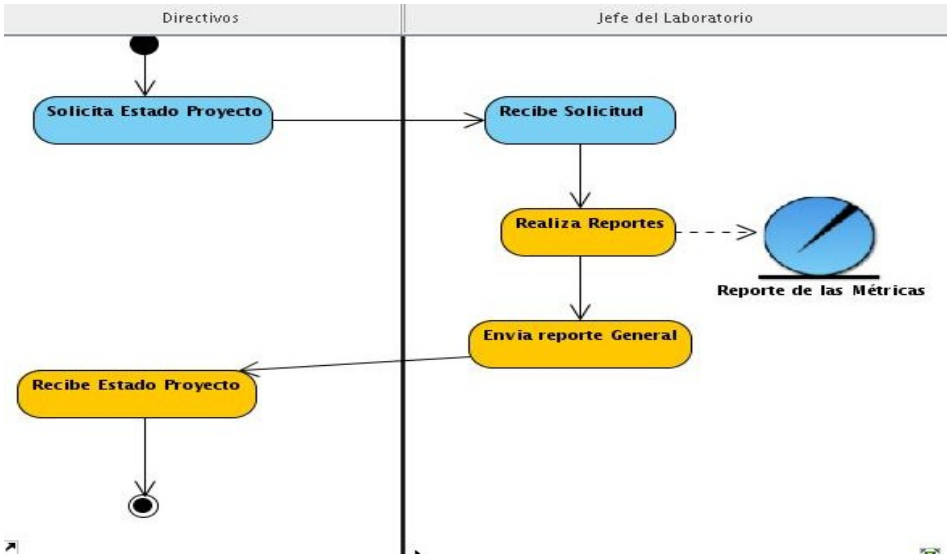
2.1.6. Descripción del negocio: CU2 Generar Reporte de Métricas.

Los directivos de la universidad le solicitan al jefe del laboratorio de pruebas el reporte estadístico de las métricas de los proyectos que se encuentran en revisión en ese momento. Luego el jefe de laboratorio, le solicita a los especialistas los reportes estadísticos de las métricas de los proyectos que se hayan revisado. Después los especialistas recogen en un reporte los resultados de las métricas aplicadas a los proyectos, ordenando los resultados por proyecto y métricas que se le calcularon. Después de confeccionar los reportes, los especialistas los envían al jefe del laboratorio. El mismo elabora un reporte general de todos los proyectos que se revisaron hasta ese momento, para luego enviárselo a los directivos de la universidad. Es necesario tener los reportes al día, pues los directivos de la universidad pueden pedirlo en cualquier momento para tener un control de cómo se encuentra el estado del proyecto según las métricas realizadas.

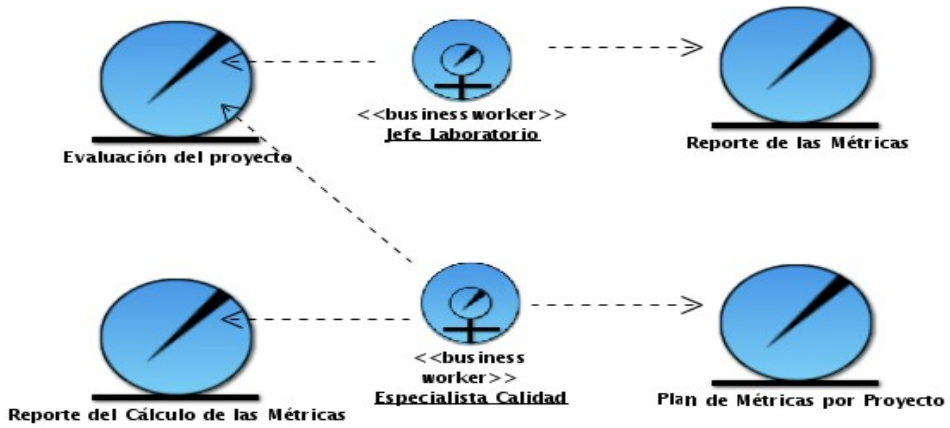
2.1.7. Diagrama de Actividades: “Gestionar Métricas”.



2.1.8. Diagrama de Actividades: “Generar Reporte de Métricas”.



2.1.9. Diagrama de Clases del Modelo de Objetos.



2.2. Definición de Requisitos Funcionales.

Los requerimientos funcionales son aquellos requisitos que, desde el punto de vista de las necesidades del usuario, son capacidades o condiciones que debe cumplir el sistema y que están fuertemente ligados a las opciones del programa. Para cumplir con los objetivos propuestos se prevé que el sistema tenga las siguientes funcionalidades:

RF 1: Gestionar Proyecto

1.1 Adicionar proyecto

1.1.1 Nombre del proyecto

1.1.2 Centro

1.1.3 Tipo de proyecto

1.2 Modificar proyecto

1.2.1 Nombre del proyecto

1.2.2 Centro

1.2.3 Tipo de proyecto

1.3 Eliminar proyecto

1.4 Visualizar proyecto

1.5 Buscar Proyecto

RF 2: Asignar métricas

2.1 Asignar métricas a un artefacto

2.1.1 Proyecto

2.1.2 Artefacto

2.1.3 Tipo de Métrica

2.1.4 Iteración

2.1.5 Métrica

2.2 Eliminar asignación de métricas a un artefacto

2.3 Visualizar asignación de métricas a un artefacto

2.4 Seleccionar métricas asignar

2.5 Buscar métrica asignada

RF 3: Calcular métricas

3.1 Insertar datos de las métricas

3.2 Calcular métricas

3.3 Visualizar cálculo de las métricas

RF 4: Imprimir Reportes de las métricas calculadas de los proyectos por tipo de características de calidad

RF 5: Evaluar Proyectos

5.1 Seleccionar proyectos

5.2 Seleccionar artefactos a tener en cuenta en la evaluación

5.3 Visualizar resultado de la evaluación

5.4 Modificar la evaluación

5.5 Buscar Proyecto evaluado

RF 6: Generar reportes de las métricas por proyecto

RF 7: Generar reporte General

RF 8: Configuración

8.1 Adicionar tipo de proyecto

8.2 Modificar tipo de proyecto

8.3 Eliminar tipo de proyecto

8.4 Visualizar tipo de proyecto

8.5 Buscar tipo de proyecto

8.6 Adicionar Centro

8.6.1 Nombre del centro

8.6.2 Especialidad del centro

8.7 Modificar Centro

8.7.1 Nombre del centro

8.7.2 Especialidad del centro

8.8 Eliminar Centro

8.9 Visualizar Centro

8.10 Buscar Centro

8.11 Adicionar tipo de artefacto

8.12 Modificar tipo de artefacto

8.13 Eliminar tipo de artefacto

8.14 Visualizar tipo de artefacto

8.15 Buscar tipo de artefacto

8.13 Adicionar artefacto.

8.13.1 Nombre del artefacto

8.13.2 Tipo de artefacto

8.14 Modificar artefacto

8.13.1 Nombre del artefacto

8.13.2 Tipo de artefacto

8.15 Eliminar artefacto

8.16 Visualizar artefacto

8.17 Buscar artefacto

8.18 Adicionar evaluación

8.19 Modificar evaluación

8.20 Eliminar evaluación

8.21 Visualizar evaluación

8.22 Buscar evaluación

8.23 Adicionar iteración

8.24 Modificar iteración

8.25 Eliminar iteración

8.26 Visualizar iteración

8.27 Buscar iteración

RF 9: Autenticar Usuario

9.1 Usuario

9.2 Contraseña

RF 10: Gestionar Roles

10.1 Adicionar roles

10.2 Modificar roles

10.3 Eliminar roles

10.4 Visualizar roles

RF 11: Gestionar Usuario

11.1 Adicionar usuario

11.2 Modificar usuario

11.3 Eliminar usuario

11.4 Visualizar Usuario

RF 12: Evaluar Artefacto

12.1 Evaluar artefacto

12.1.1 Proyecto

12.1.2 Artefacto

12.1.3 Iteración

12.1.4 Evaluación

12.2 Seleccionar métricas a tener en cuenta en la evaluación

12.3 Visualizar resultado de la evaluación

12.4 Modificar la evaluación

12.5 Buscar artefacto evaluado

2.3. 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 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.

1. Apariencia

El sistema debe poseer una interfaz Web sencilla, amigable y clara para el usuario, además su funcionamiento debe ser de fácil comprensión.

2. Usabilidad

El sistema solo puede ser utilizado por aquellas personas que se encuentren relacionadas con las pruebas de liberación de los proyectos productivos de la Universidad de las Ciencias Informáticas, en el Laboratorio Industrial de Pruebas de Software. Por ejemplo el especialista de calidad y el jefe del laboratorio.

3. Rendimiento

Se obtendrán buenos resultados con poco trabajo. Debido a que si se trata de una aplicación cliente/servidor la misma debe ser eficiente, con capacidad adecuada de procesamiento y cálculo, así como requiere de un tiempo de respuesta relativamente pequeño.

4. Portabilidad

El sistema debe ser multiplataforma.

5. Seguridad

Permitir mantener la seguridad del sistema. Para poder acceder al sistema el usuario deberá estar registrado en la aplicación. Los usuarios serán creados en dependencia del rol que desempeñen. Se debe garantizar que sólo posean acceso a la información con derecho a ver o manipular según el rol.

6. Software

El software debe poseer métodos de instalación rápidos, debe facilitar también la reinstalación en caso de contingencia. En el lado del Cliente debe existir un navegador que soporte Adobe Reader. En el lado del Servidor debe estar instalado PHP 5.0 o superior a este y cómo gestor de base de datos PostgreSQL.

2.4. Actores del Sistema a Automatizar.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato para actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema. Los actores del sistema se clasifican por:

- No son parte del sistema.
- Pueden intercambiar información con el sistema.
- Pueden ser un recipiente pasivo de información.
- Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado.

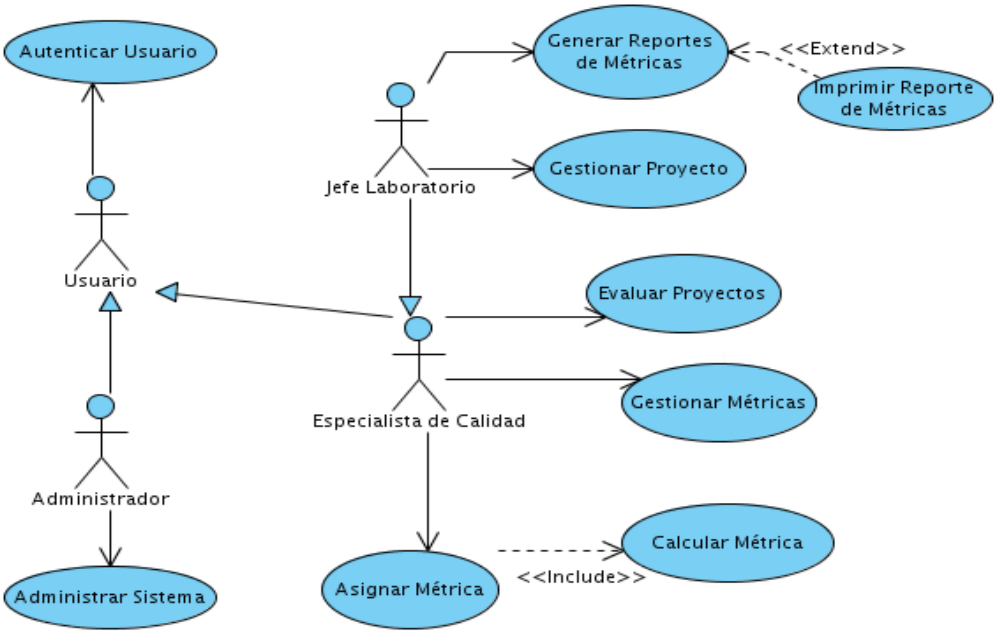
Tabla 3: Actores del sistema.

Actor	Descripción
Jefe del Laboratorio	Encargado de elaborar los reportes de métricas de todos los proyectos que se están revisando, para luego enviárselos a los directivos de la universidad.
Especialista de Calidad	Encargado de dirigir las pruebas de liberación que se le realicen a los proyectos, así como de calcular las métricas asociadas a los proyectos.
Administrador	Encargado del mantenimiento del sistema así como de gestionar todo el proceso de permisos a los usuarios que acceden al mismo.
Usuario	Actor global que permite la autenticación de los diferentes usuarios.

2.5. Descripción del Diagrama de Casos de Uso del Sistema.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores. Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, así como el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

2.6. Diagrama de Casos de Uso del Sistema.



2.7. Descripción de los Casos de Uso del Sistema.

Tabla 4: Descripción CU Asignar Métricas.

Caso de Uso:	Asignar Métricas
Actores:	Especialista de Calidad
Resumen:	El caso de uso inicia cuando el Especialista de Calidad asigna las métricas que se le calcularán a los proyectos que están en pruebas.
Precondiciones:	El actor debe haberse autenticado como Especialista de Calidad.
Referencias	**RF2.**
Prioridad	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción de asignar métricas al proyecto.	2. El sistema muestra la interfaz con los proyectos asignados al Especialista de Calidad. Permite asignar métricas a los proyectos.
3. El actor escoge un proyecto para asignarle métricas y selecciona la opción Asignar Métricas.	4. El sistema muestra la interfaz para asignar las métricas al proyecto.
5. El actor introduce los datos y selecciona la opción Asignar.	6. El sistema asigna las métricas al proyecto. Permite calcular las métricas asignadas al proyecto. Ver CU Calcular Métricas. El caso de uso termina.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El actor selecciona la opción Cancelar.	* El sistema cancela la operación y muestra la vista anterior.
Prototipo de Interfaz	
Poscondiciones	Quedan asignadas las métricas a los proyectos.

Tabla 5: Descripción CU Calcular Métricas.

Caso de Uso:	Calcular Métricas.
Actores:	Especialista de Calidad.
Resumen:	El caso de uso inicia cuando el Especialista de Calidad calcula las métricas asignadas a los proyectos que están en pruebas en ese momento.
Precondiciones:	El actor debe estar autenticado como Especialista de Calidad y haberse asignado métricas a los proyectos.
Referencias	**RF3.**
Prioridad	
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El actor selecciona la opción Calcular Métricas.	2. El sistema muestra la interfaz para el cálculo de las métricas.
3. El actor introduce los datos y selecciona la opción Calcular.	4. El sistema valida los datos y realiza el cálculo de las métricas. El caso de uso termina.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El actor selecciona la opción Cancelar.	* El sistema cancela la operación y muestra la vista anterior.
3.1. El actor deja campos vacíos y selecciona la opción Calcular.	3.2. El sistema muestra un mensaje de información: “Debe llenar todos los campos”. Regresa al paso 3 del flujo básico.
3.3. El actor introduce datos erróneos y selecciona la opción Calcular.	3.4. El sistema muestra un mensaje de información: “Existen errores en los datos”. Regresa al paso 3 del flujo básico.
Prototipo de Interfaz	
Poscondiciones	Quedan calculadas las métricas.

Tabla 6: Descripción CU Evaluar Proyectos.

Caso de Uso:	Evaluar Proyectos
Actores:	Especialista de Calidad
Resumen:	El caso de uso inicia cuando el Especialista de Calidad decide evaluar los proyectos.
Precondiciones:	El actor debe estar autenticado como Especialista de Calidad y haberse calculado las métricas.
Referencias	**RF5.**
Prioridad	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El actor selecciona la opción Evaluar Proyecto.	2. El sistema muestra la interfaz con los proyectos asignados al Especialista de Calidad. Permite evaluar los proyectos.
3. El actor escoge el proyecta a evaluar y selecciona la opción Evaluar.	4. El sistema evalúa el proyecto por los criterios de evaluación previamente definidos. El caso de uso termina.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El actor selecciona la opción Cancelar.	* El sistema cancela la operación y muestra la vista anterior.
Prototipo de Interfaz	
Poscondiciones	Quedan evaluados los proyectos.

Se representa la descripción de los CU restantes en el Anexo 1.

2.8. Conclusiones del Capítulo.

En este capítulo se dio una breve descripción acerca de los roles que desarrollan dichos procesos y de otras personas que puedan estar involucradas. Se definió además la información manejada y la propuesta del sistema. Se identificaron los requisitos funcionales y no funcionales, los casos de uso con sus descripciones, los diagramas de actividades y el modelo de objeto.

Capítulo 3: Análisis y Diseño del sistema.

En este capítulo se dará una breve descripción del análisis de los casos de uso del sistema para diseñar las clases que se implementarán. Se representan además los diagramas de colaboración del diseño, el diagrama de las clases diseñadas con sus relaciones, los principios utilizados para el diseño de dichas clases, los diagramas de clases persistentes, el diagrama entidad relación y la descripción de las tablas de la base de datos.

3.1. Modelo de Análisis.

El Modelo de Análisis está estructurado mediante una jerarquía de paquetes del análisis que contiene diagramas de clases del análisis y diagramas de secuencia, que describen la aplicación lógica de los requisitos funcionales que se identificaron en el modelo de casos de uso. El modelo de análisis identifica las principales clases en el sistema, y contiene relaciones de casos de uso que describen como el sistema se construirá. Los diagramas de secuencia describen el flujo de eventos de los casos de uso al ser ejecutados. El modelo de análisis es la base del modelo de diseño, pues describe la estructura lógica del sistema, pero no como se llevará a cabo.

3.1.1. Diagramas de Clases del Análisis.

Las clases de análisis representan una abstracción de los subsistemas del diseño del sistema, y se centran en los requisitos funcionales. Las clases del análisis siempre encajan en uno de los tres estereotipos básicos: de interfaz, de control o entidad. Cada estereotipo implica una semántica específica, lo cual constituye un método potente y consistente de identificar y describir las clases del análisis.

A continuación se muestran los diagramas de clases del análisis:

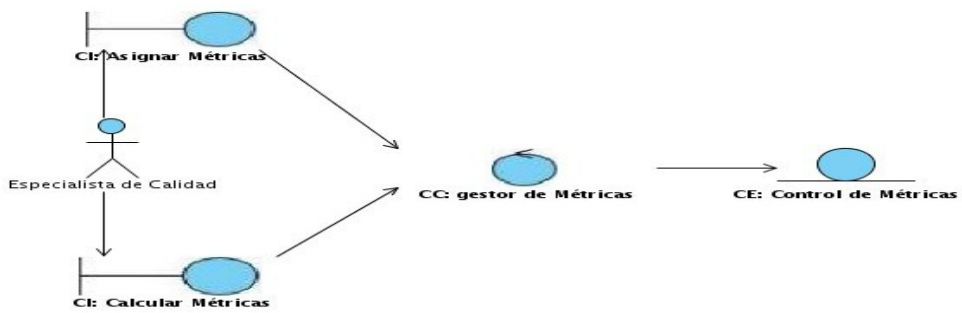


Figura 3: Diagrama de Clase del Análisis: Asignar y Calcular Métricas.

Se representan los restantes diagramas de clases del análisis en el Anexo 2.

3.2. Modelo del Diseño.

El Modelo del Diseño es un modelo de objetos que describe la realización física de 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. Una clase de diseño representa una abstracción del subsistema de la implementación del sistema. Esta abstracción es sin costuras debido a (Jacobson et al., 1999):

- El lenguaje utilizado para especificar una clase de diseño es el mismo que el lenguaje de programación utilizado.
- Se especifica la visibilidad de atributos y operaciones, utilizando con frecuencia los términos derivados de C++.
- Las relaciones entre clases de diseño suelen tener un significado directo cuando la clase es implementada.
- Los métodos de una clase de diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.
- Una clase de diseño puede aparecer como un estereotipo que se corresponde con una construcción en el lenguaje de programación dado.
- Una clase de diseño se puede realizar, esto es, proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación.
- Una clase de diseño se puede activar, implicando que objetos de la clase mantengan su propio hilo de control y se ejecuten concurrentemente con otros objetos activos.

3.2.1. Diagramas de Colaboración del Diseño.

El diagrama de colaboración es una forma alternativa de mostrar un escenario. Este diagrama muestra las interacciones entre objetos organizadas entorno a los objetos y los enlaces entre ellos. Los diagramas de colaboración brindan la representación principal de un escenario, ya que las colaboraciones se organizan entorno a los enlaces de unos objetos con otros.

A continuación se muestran los diagramas de colaboración del diseño:

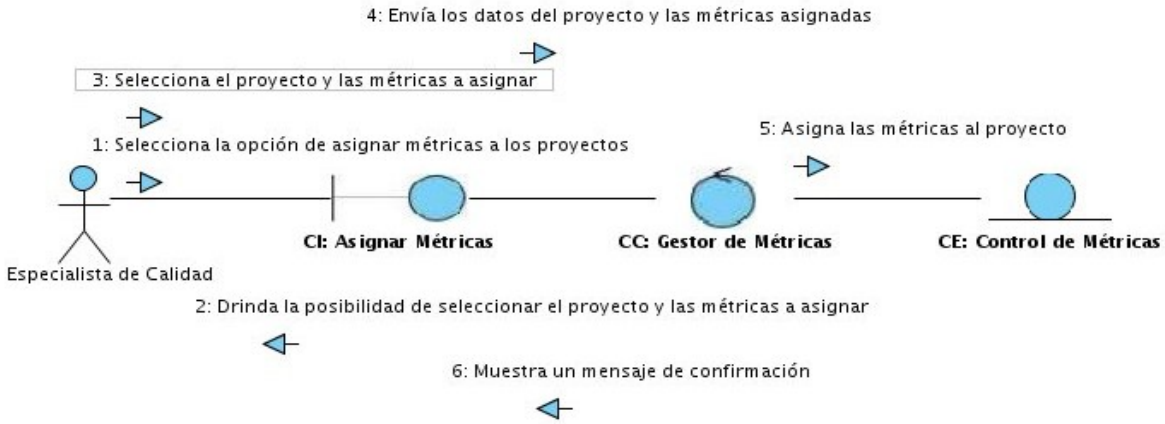


Figura 4: Diagrama de Colaboración del Diseño: Asignar Métricas.

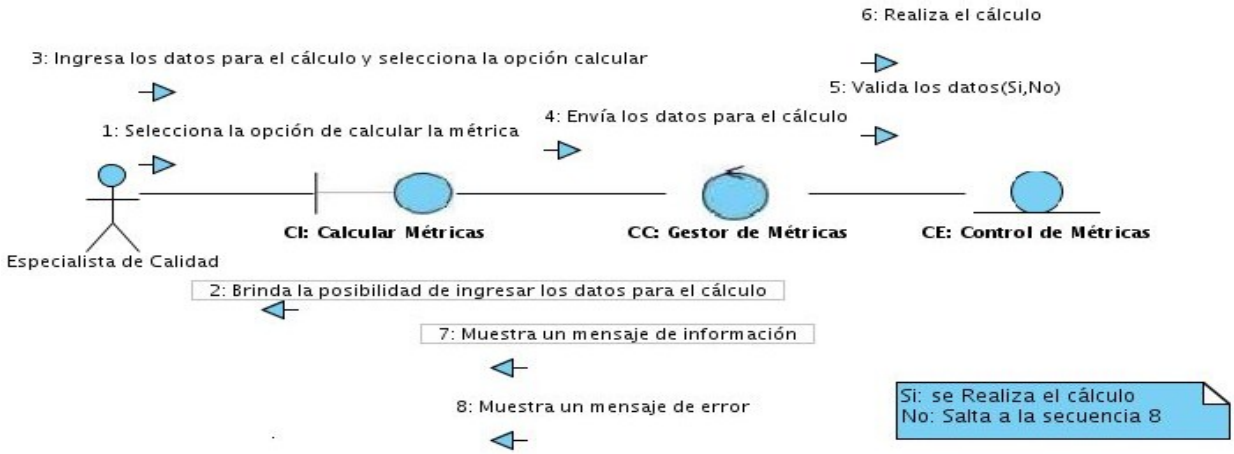


Figura 5: Diagrama de Colaboración del Diseño: Calcular Métricas.

Se representan los restantes diagramas de colaboración del diseño en el Anexo 3.

3.2.2. Diagrama de Clases del Diseño.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real. Normalmente contiene clases, asociaciones, interfaces y métodos.

A continuación se muestran los diagramas de colaboración del diseño:

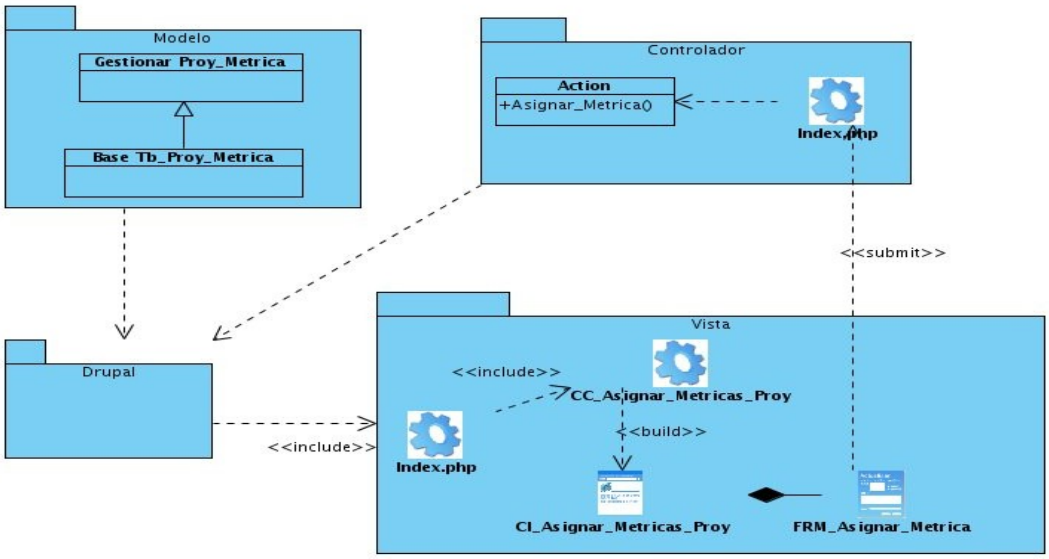


Figura 6: Diagrama de Clases del Diseño: Asignar Métricas.

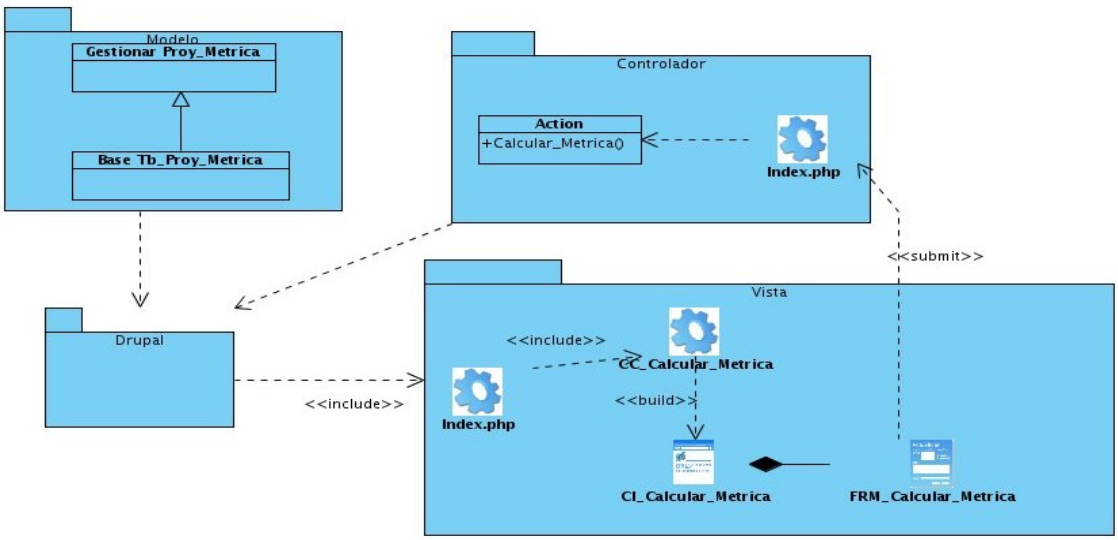


Figura 7: Diagrama de Clases del Diseño: Calcular Métricas.

Se representan los restantes diagramas de clases del diseño en el Anexo 4.

3.3. Arquitectura Web en Capas.

La arquitectura de las aplicaciones Web suelen presentar un esquema basado en capas. Este sistema en capas tiene una organización jerárquica, donde cada capa proporciona servicios a la

capa superior y sirve de cliente a la capa inferior.

Este tipo de arquitectura facilita la migración, pues el acoplamiento con el entorno se encuentra en las capas inferiores y serían las únicas a reimplementar en caso de transporte a otro entorno. Otra de las ventajas es la reutilización, al estar implementadas en cada nivel unas interfaces claras se pueden intercambiar. Como los cambios en una capa afecta muy poco a las demás, es relativamente fácil de darle mantenimiento. Por otro lado no todos los sistemas se pueden estructurar fácilmente en capas. También tiene la desventaja de que la comunicación entre capas puede hacer ineficiente el sistema.

3.3.1. Patrón arquitectónico en Drupal.

Drupal para conseguir flexibilidad y facilidad en la creación de sitios web se basa en la organización en capas que aplica al tratar los contenidos. En lugar de tratar al sitio como un conjunto de páginas, Drupal estructura los contenidos en una serie de elementos básicos, que son los nodos, módulos, bloques y menús, permisos de usuarios y plantillas.

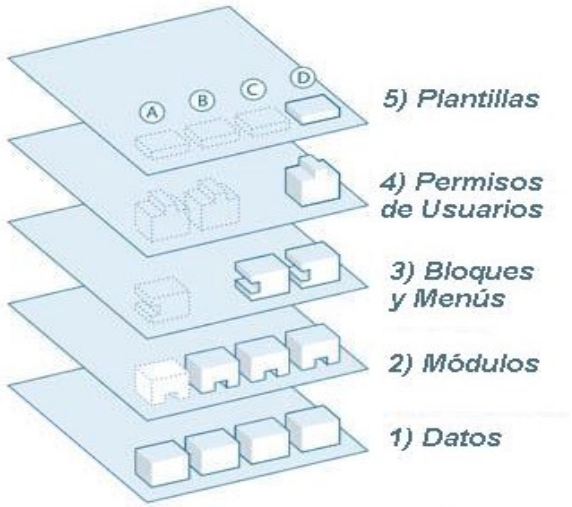


Figura 8: Arquitectura en capas de Drupal.

Los nodos son los elementos básicos en que Drupal almacena la información. Así los nodos crecen a medida que el sitio web crece. Los nodos van formando un “depósito de nodos” cada vez mayor. Se puede decir que la primera capa de la estructura de Drupal la forma este “deposito” de nodos.

Los módulos operan sobre los nodos y otorgan funcionalidad a Drupal permitiendo incrementar sus capacidades o adaptarlas a las necesidades de cada sitio web. Son una especie de Plug-Ins

que se instalan en el sitio web proporcionándole nuevas funcionalidades.

La siguiente capa de Drupal la constituyen los bloques y menús. Estos permiten estructurar y organizar los contenidos en la página web. O sea, son los elementos que albergan y permiten acceder al usuario a la salida generada y procesada por los módulos a partir de la información almacenada en los nodos.

La siguiente capa importante en Drupal es la de control de usuarios y permisos. Actualmente, la mayor parte de sitios web son multiusuario, por lo que la seguridad y control de los usuarios es un punto clave para garantizar la integridad de la información almacenada. Drupal dispone de un registro de usuarios y de roles que permiten especificar que tareas pueden realizar y a que contenidos puede acceder cada tipo de usuario. Es decir que las operaciones que se pueden realizar sobre los elementos provenientes de las capas inferiores se encuentran limitadas por la capa de control de usuarios y permisos de Drupal.

La última capa, es la capa de temas y es la que establece la apariencia gráfica o estilo de la información que se le muestra al usuario. Esta separación entre información y aspecto gráfico permite cambiar el diseño del sitio web sin necesidad de modificar los contenidos.

3.4. Diseño de la Base de Datos

Las bases de datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. Es por ello que su diseño es muy importante en su desarrollo. Al diseñar una base de datos, se refleja la estructura del problema en el mundo real, evita el almacenamiento de información redundante y permite además representar todos los datos esperados, incluso con el paso del tiempo. Para lograr diseñar la base de datos se partió de la definición de las clases persistentes, lo cual podemos definir como la capacidad de un objeto de mantener su valor en el espacio y tiempo. Después del refinamiento y clasificación de estas clases y sus atributos se realizó la construcción del diagrama de clases persistentes.

3.4.1. Modelo lógico de datos

El modelo lógico de datos representa la información que maneja el sistema, siendo una fuente de información para el modelo físico. A continuación se muestra el Modelo lógico de datos diseñado para la base de datos de la aplicación de este trabajo de diploma, el cual no es más que el diagrama de clases persistentes. El diagrama de clases persistentes es un diagrama de clases en el que, como bien indica su nombre, solo aparecen las clases persistentes, de las que hay que

expandir detalles estructurales. Estas clases persistentes tuvieron como origen las clases clasificadas como entidad, ya que estas modelan la información y el comportamiento asociado al fenómeno en cuestión.

A continuación se muestra el diagrama de clases persistentes:

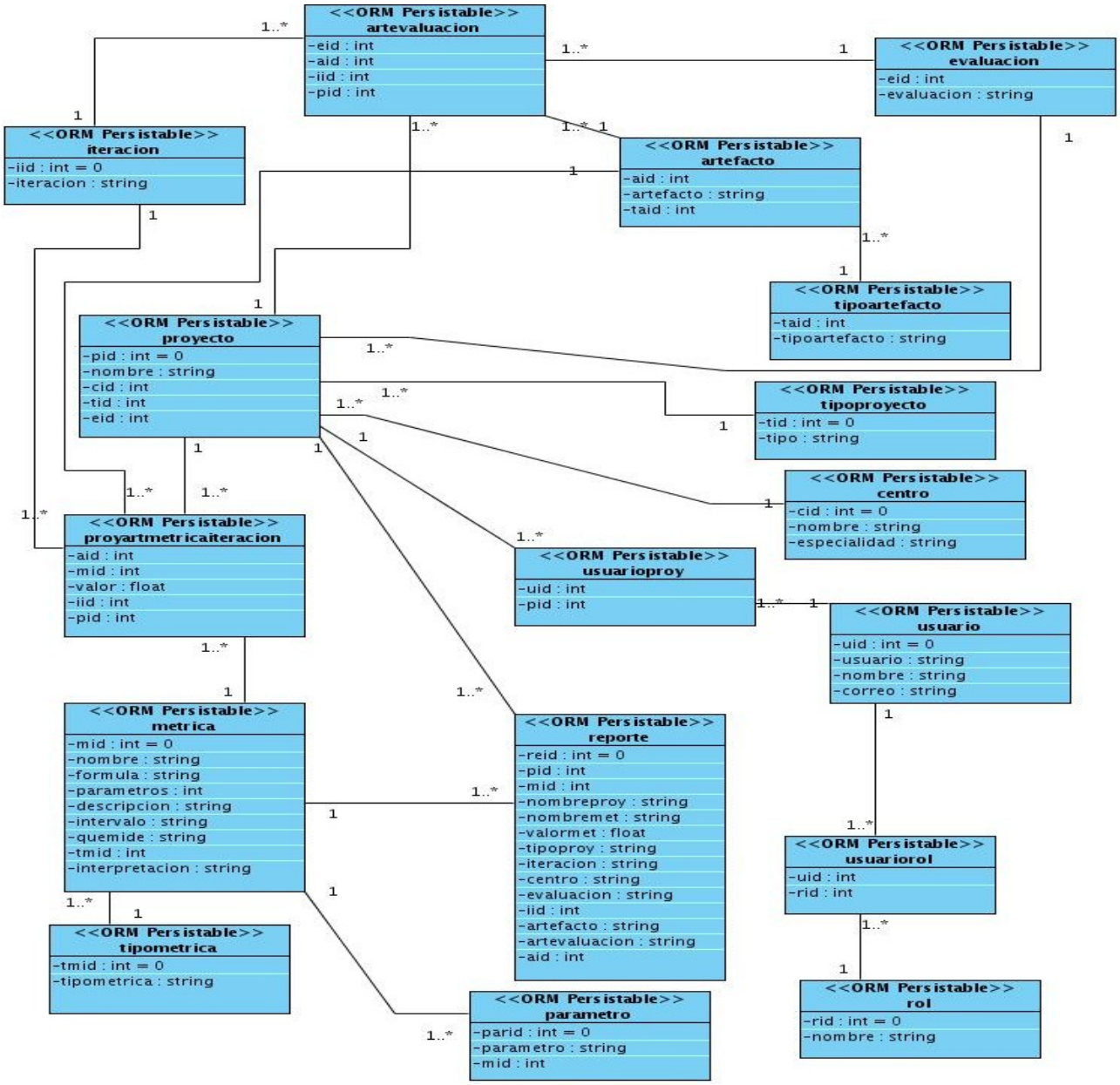


Figura 9: Diagrama de Clases Persistentes.

3.5. Diseño de la Base de Datos. Modelo Entidad Relación.

El modelo entidad relación es una herramienta para el modelado de datos. En estos modelos se representan entidades relevantes con sus interrelaciones y propiedades.

A continuación se muestra el diagrama de entidad relación:

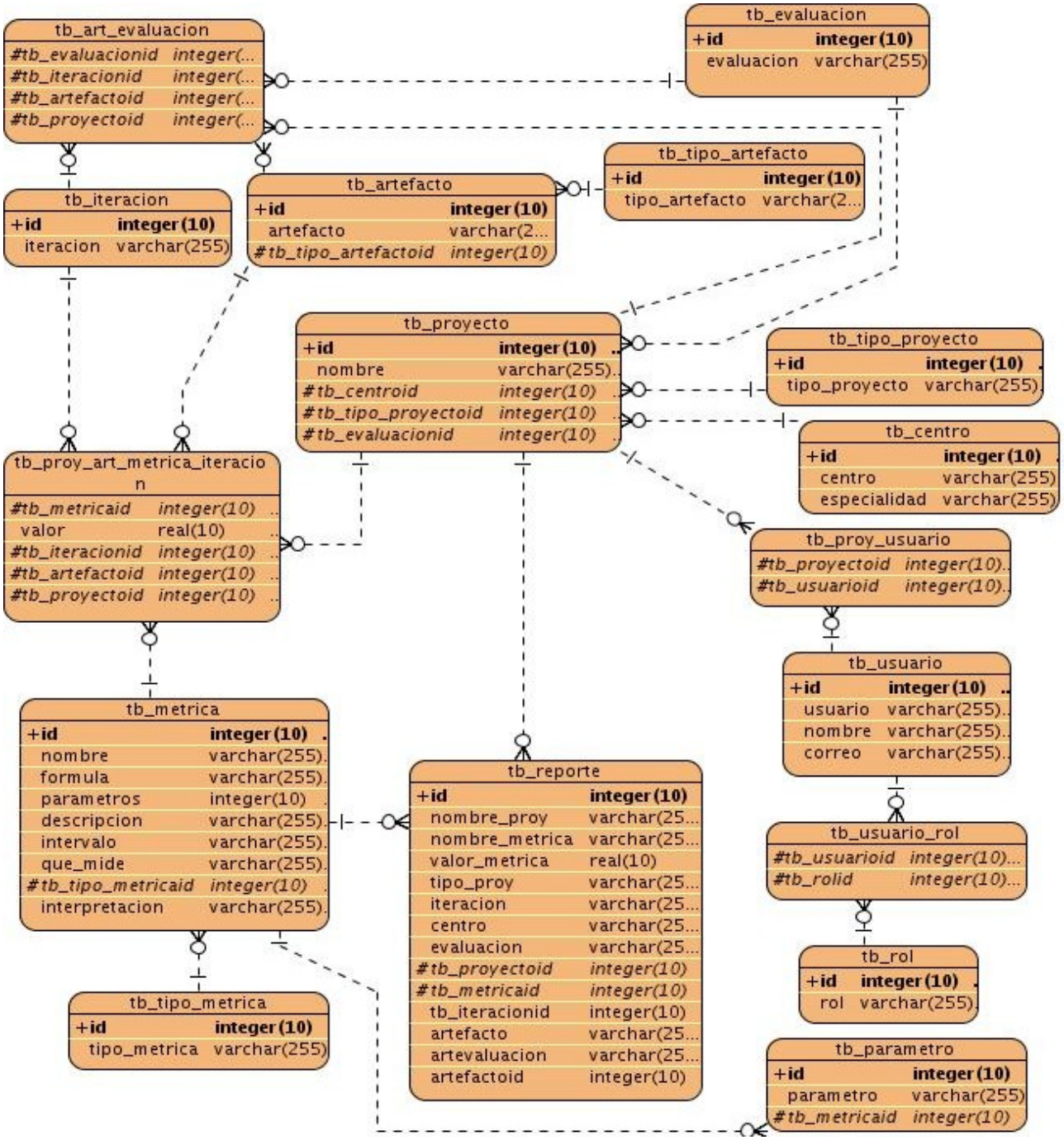


Figura 10: Modelo Entidad Relación.

3.5.1. Descripción de las Tablas.

En las descripciones de las tablas se especifica la función de cada tabla dentro del modelo de datos. Así como describir las particularidades de sus atributos.

Se representa la descripción de todas las tablas de la Base de Datos en el Anexo 5.

3.6. Principios del Diseño.

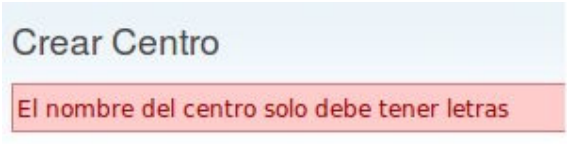
Para garantizar un vínculo entre el usuario y el sistema llevado a cabo se ha hecho un esfuerzo por garantizar la mejor claridad posible en la interfaz procurando que sea intuitiva y que garantice una funcionalidad óptima del sistema y una comodidad en el trabajo de los usuarios finales.

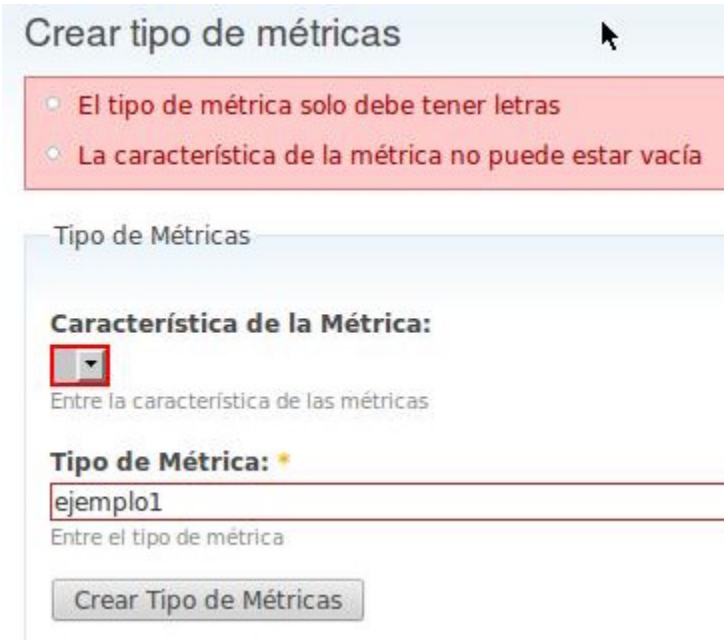
Se trabajó con formularios Web, para lograr una mayor visualización de la información, se utilizó un mismo color en todas las páginas y un mismo tipo y tamaño de letra con textos claros, el color predominante es el azul. El envío de la información debe ser lo más rápida posible por lo que no se utilizaron muchas imágenes ni funciones que atenten contra esto y que posibiliten además una navegación rápida y eficiente.

3.7. Tratamiento de Errores.

Se contará con un sistema de tratamiento de errores para disminuir la posibilidad de cometerlos. Para esto se detallará con la validación de la información introducida en el sistema. Mediante la interfaz Web se evitará que el usuario asuma un papel activo en la introducción de la información, para esto se contará con cuadros de opción, menú de selección lo cual facilitará la entrada de datos. La información que pretenda ser adicionada por el usuario se validará mediante funciones que avalen que sea válida y que el cuadro de texto no se encuentre vacío si es obligatorio llenarlo. Si hay un error en la información le saldrá al usuario un mensaje en pantalla indicándole el error.

También se validarán las opciones correspondientes a la extracción o modificación de datos del servidor de base datos. Si se desea eliminar algún elemento de la base de datos se preguntará al usuario si está seguro de realizar dicha acción, al igual que cuando desee modificar alguna información, antes de actualizarla se le preguntará si desea realizarla o no. Así se logra que se ejecuten las operaciones que se desean y que se rectifique al cometer un error. A continuación se muestran algunos de estos mensajes:





3.8. Seguridad.

La seguridad en el sitio está implementada a través del módulo user, que verifica que el usuario existe en la aplicación y que los datos que introduce sean correctos, en caso de no ser así no tendrá acceso a la aplicación. Para garantizar la seguridad de la información se crearon varios niveles de seguridad, definidos como roles de usuario, que pueden ser: especialista de calidad, jefe del laboratorio de prueba y el administrador. Este último es el encargado del buen funcionamiento del sistema por lo que tendrá derecho al control total del mismo. Los demás usuarios no tendrán acceso a la información restringida para ellos, para esto se trabaja con una lista de acceso que contiene los diferentes permisos que pueden tener los diferentes roles. A continuación se muestran algunos de los mensajes que son mostrados en caso de violar las normas de seguridad:

3.9. Conclusiones del Capítulo.

En este capítulo se dio una breve explicación acerca del análisis de los casos de uso del sistema para diseñar las clases que se implementarán, se representaron los diagramas de colaboración del diseño, los diagramas de las clases diseñadas con sus relaciones, los principios utilizados para el diseño de dichas clases, los diagramas de clases persistentes, el diagrama entidad relación y la descripción de las tablas de la base de datos. También se explicó la definición del diseño que se aplicó, la forma de tratar los errores y la seguridad del sistema.

Capítulo 4. Implementación y Prueba.

En este capítulo se reflejan los diferentes diagramas correspondientes a este flujo de trabajo, donde se implementará la propuesta realizada en el análisis y diseño. Se representa el diagrama de despliegue, los de componentes y el modelo de prueba correspondiente a la aplicación desarrollada.

4.1. Implementación.

En la implementación se comienza con el resultado del análisis y diseño, y se implementa el sistema en términos de componentes, es decir: ficheros de código fuente, scripts, ficheros de códigos binarios y ejecutables. Dentro de los principales objetivos se encuentran: distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue e implementar las clases y subsistemas encontrados en el diseño.

4.1.1. Diagrama de Despliegue.

El modelo de despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de 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.

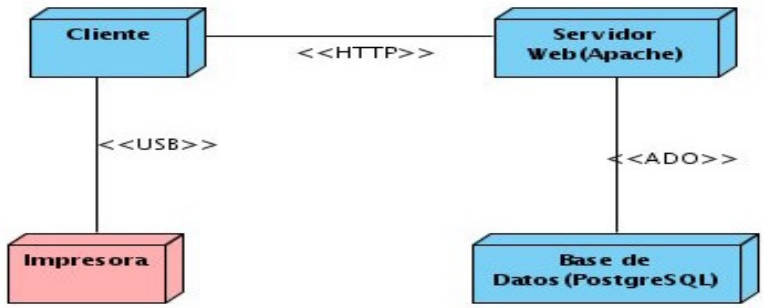


Figura 11: Diagrama de Despliegue.

4.1.2. Diagramas de Componentes.

El diagrama de componente se encuentra presente en el diseño que da paso a la implementación, y se define a partir del diagrama de clases. Para su elaboración se deben identificar todas las

clases que participan en el sistema, luego se identifican los métodos quienes pasan a ser módulos con líneas de código independientes. Después los módulos serán componentes que se relacionan entre si.

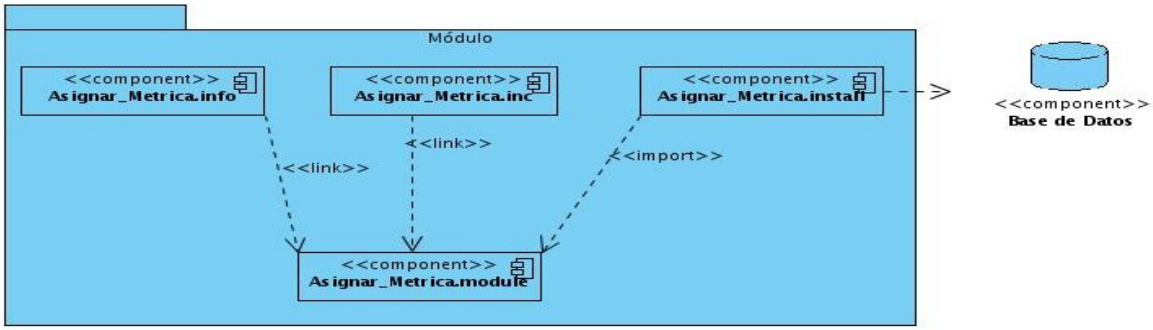


Figura 12: Diagrama de Componente Asignar Métrica.

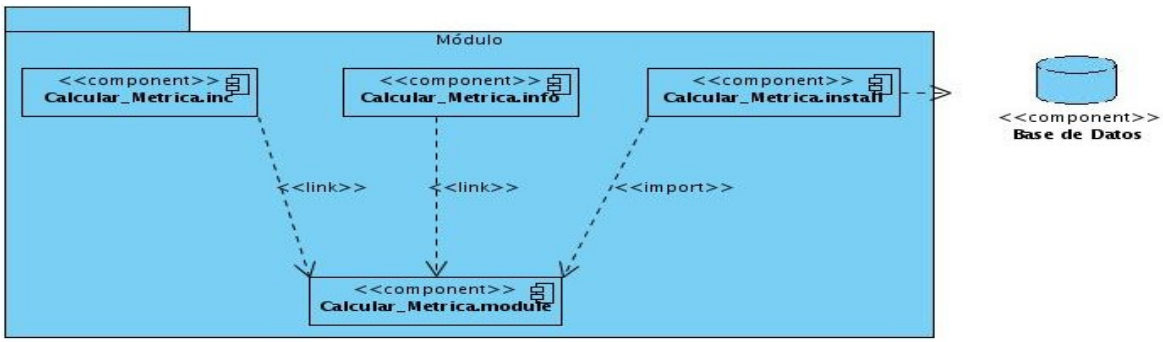


Figura 13: Diagrama de Componente Calcular Métrica.

Se representan los restantes diagramas de componente en el Anexo 6.

4.2. Pruebas.

Las pruebas de software son el instrumento adecuado para conocer el estado en que se encuentra la calidad del software. En este proceso se realizan pruebas dirigidas a partes del software o a todo el sistema, con el objetivo de medir el grado en que el software cumple con los requerimientos.

4.2.1. Técnicas de Pruebas.

Existen diferentes técnicas de pruebas entre las que se encuentran:

Técnica de Caja Blanca: Basada en los requerimientos, en el diseño y en el código fuente.

- ✓ **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.

Técnica de Caja Negra: Es basada en los requerimientos y verifica si el sistema realmente hace lo que debe hacer, es decir, se llevan a cabo sobre la interfaz del software. 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.

4.2.2. Tipos de Pruebas.

Entre los tipos de prueba que existen se encuentran los siguientes:

Prueba de Recuperación: Fuerza un fallo del software y verifica que la recuperación se lleva a cabo apropiadamente.

Prueba de Seguridad: Prueba centrada en asegurar que los datos o sistemas que son objetos de prueba, son accedidos sólo por los actores que tienen permiso para hacerlo. Esta prueba es implementada y ejecutada contra varios objetos de prueba.

Prueba de Resistencia: Enfrenta a los programas a situaciones anormales.

Prueba de Rendimiento: Prueba el rendimiento del software en tiempo de ejecución.

Prueba de Instalación: Se centra en asegurar que el sistema de software desarrollado se puede instalar en diferentes configuraciones, hardware y software, bajo condiciones y excepciones, por ejemplo con espacio de disco insuficiente o continuas interrupciones.

Prueba función: Es la prueba centrada en validar las funciones que son objeto de prueba como lo que deben ser, ofreciendo los servicios, métodos o casos de usos requeridos. Esta prueba es implementada y ejecutada contra diferentes objetos de pruebas, incluyendo unidades, unidades integradas, aplicaciones y sistemas.

Se le aplican a todas o solo un subconjunto de pruebas de función. Puede implicar la re-ejecución de cualquier tipo de prueba. Normalmente, esta prueba se lleva a cabo durante cada iteración, ejecutando otra vez la prueba de la iteración anterior.

Prueba Usabilidad: Prueba encaminada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.

Prueba de Integridad: Enfocada a la valoración de la robustez (resistencia a fallos).

Prueba de Stress: Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible).

Prueba de Carga: Es la prueba usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema está bajo prueba permanece constante. La variación en carga es simular a la carga de trabajo promedio y con picos que ocurren dentro de tolerancias operacionales normales. (Pressman, 1990)

4.2.3. Niveles de Pruebas.

Entre los niveles de pruebas prevalecen:

Pruebas de Unidad: Es realizada por el desarrollador y se centra en el módulo. Utilizando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. La prueba de unidad hace uso intensivo de las técnicas de prueba de caja blanca.

Pruebas de Integración: Es la que verifica si el conjunto de componentes que conforma al sistema completo interactúa correctamente. Se recomienda que esta fase sea llevada a cabo por una entidad distinta al grupo de desarrollo por cuestiones de eficiencia y objetividad. Tiene como objetivo obtener los módulos probados en la prueba de unidad y construir una estructura del programa, que esté de acuerdo con lo que dicta el diseño.

Existen dos formas de integración:

- ✓ **Integración no incremental**: Se combinan todos los módulos por anticipado y se prueba todo el programa en conjunto.

- ✓ **Integración incremental**: El programa se construye y se prueba en pequeños segmentos.

En la prueba de integración el foco de atención es el diseño y la construcción de la arquitectura del software.

Las técnicas que más prevalecen son las de diseño de casos de prueba de caja negra, aunque se pueden llevar a cabo unas pocas pruebas de caja blanca.

Pruebas de Sistema: Verifica que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total. La prueba del sistema está constituida por una serie de pruebas diferentes, cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. (Profesor: López.)

4.2.4. Estrategia de Prueba.

La estrategia de prueba que se va a implementar a esta aplicación es la que se muestra a continuación:

- ✓ **Pruebas de Integración**. A partir del esquema del diseño, los módulos probados se combinan para comprobar que funcionan juntos.
- ✓ **Pruebas de Sistema**. El software ensamblado totalmente con cualquier componente hardware que requiere de prueba para comprobar que se cumplen los requisitos funcionales.

4.2.5. Casos de Prueba.

El objetivo de un caso de prueba es especificar una manera de probar el sistema, incluyendo las entradas con las que se va a probar, los resultados esperados y las condiciones bajo las que se ha de probar. Los casos de prueba son un producto de desarrollo de software que sirven para comprobar las expectativas de los stakeholders. Generalmente un caso de prueba se deriva de un caso de uso o de una realización de un caso de uso en el modelo de diseño. Un caso de prueba basado en un caso de uso especifica una prueba del sistema como caja negra, o sea, una prueba observable externamente del sistema.

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

Tabla 7: Secciones a probar en el Caso de Uso Asignar Métrica.

Nombre de la Sección	Escenarios de la Sección	Descripción
SC1: Asignar Métrica.	EC 1.1: Introduce los datos de la Asignación. Selecciona la opción de asignar.	Valida los datos. Guarda la Asignación. Muestra un mensaje de información.
	EC1.2: Existen campos incompletos.	No se asigna la métrica. Muestra un mensaje de información.
	EC1.3: Existen campos incorrectos.	No se asigna la métrica. Muestra un mensaje de información.
SC2: Eliminar Asignación de una Métrica.	EC2.1: Selecciona la opción de eliminar una Métrica Asignada.	Muestra un mensaje de advertencia. Y permite: Eliminar. Cancelar.
	EC2.2: Selecciona la opción Eliminar.	Elimina la asignación de la Métrica.
	EC2.3: Selecciona la opción de Cancelar.	No elimina la asignación. Regresa a la vista anterior.

Tabla 8: Secciones a probar en el Caso de Uso Evaluar Proyecto.

Nombre de la Sección	Escenarios de la Sección	Descripción
SC1: Evaluar Proyecto.	EC 1.1: Introduce los datos de la Evaluación. Selecciona la opción de evaluar un Proyecto.	Valida los datos. Guarda la Evaluación. Muestra un mensaje de información.
	EC1.2: Existen campos incompletos.	No se evalúa el proyecto. Muestra un mensaje de información.
	EC1.3: Existen campos incorrectos.	No se evalúa el proyecto. Muestra un mensaje de información.

SC2: Eliminar Evaluación de un Proyecto.	EC2.1: Selecciona la opción de Eliminar la evaluación de un proyecto.	Muestra un mensaje de advertencia. Y permite: Eliminar. Cancelar.
	EC2.2: Selecciona la opción Eliminar.	Elimina la evaluación del proyecto.
	EC2.3: Selecciona la opción de Cancelar.	No elimina la evaluación de proyecto. Regresa a la vista anterior.
SC3: Modificar Evaluación de un Proyecto.	EC3.1: Modifica los datos que necesite y selecciona la opción de actualizar los datos.	Actualiza los datos de la evaluación del proyecto. Muestra un mensaje de información.
	EC3.2: Existen datos incompletos.	No modifica la evaluación del proyecto. Muestra un mensaje de información.

Tabla 9: Secciones a probar en el Caso de Uso Gestionar Proyecto.

Nombre de la Sección	Escenarios de la Sección	Descripción
SC1: Incluir Proyecto.	EC1.1: Introduce los datos del Proyecto. Selecciona la opción de incluir un Proyecto.	Valida los datos. Guarda el Proyecto. Muestra un mensaje de información.
	EC1.2: Existen campos incompletos.	No se incluye el proyecto. Muestra un mensaje de información.
	EC1.3: Existen campos incorrectos.	No se incluye el proyecto. Muestra un mensaje de información.
SC2: Eliminar Proyecto.	EC2.1: Selecciona la opción de Eliminar un proyecto.	Muestra un mensaje de advertencia. Y permite: Eliminar. Cancelar.

	EC2.2: Selecciona la opción Eliminar.	Elimina el proyecto y los elementos relacionados al proyecto.
	EC2.3: Selecciona la opción de Cancelar.	No elimina el proyecto. Regresa a la vista anterior.
SC3: Modificar Proyecto.	EC3.1: Modifica los datos que necesite y selecciona la opción de actualizar los datos.	Actualiza los datos del proyecto. Muestra un mensaje de información.
	EC3.2: Existen datos incompletos.	No modifica el proyecto. Muestra un mensaje de información.
	EC1.3: Existen campos incorrectos.	No modifica el proyecto. Muestra un mensaje de información.

4.2.6. Resultado de las Pruebas.

En las pruebas funcionales realizadas a la herramienta se obtuvieron 35 No Conformidades en las dos iteraciones realizadas, de ellas 34 fueron significativas y 1 no significativa. De las No Conformidades encontradas 7 fueron de validación, 27 de opciones que no funcionan y 1 de ortografía. Los resultados se muestran en la siguiente tabla.

Tabla 10: Resultado de las pruebas.

No Conformidades	1ra Iteración	2da A Iteración
No Significativas	0	0
Recomendaciones	0	0
Formato	0	0
Error técnico	0	0
Validaciones	7	0
Opciones que no funcionan	27	0
Errores de interfaz	0	0

Funcionalidades	0	0
Excepciones	0	0
Ortografía	1	0
Redacción	0	0
Errores de idioma	0	0
Otras documentaciones	0	0

4.3. Conclusiones del Capítulo.

- ✓ En este capítulo se realizó toda la implementación del sistema y se mostraron los diagramas de despliegue y componentes.
- ✓ Con los casos de pruebas elaborados se comprobó la integración del módulo al sistema.
- ✓ A partir del diseño de clases persistentes y la estructuración del diagrama de clases persistentes se obtuvo el modelo de datos, lo que facilitó el diseño de las tablas de la base de datos.
- ✓ La realización del diagrama de despliegue propició una visión de cómo está distribuido el sistema físicamente.

Conclusiones.

- ✓ La elaboración de la herramienta informática, implementada a partir de la necesidad de automatizar los procesos que se llevan a cabo en el Laboratorio de Pruebas de CALISOFT, permitirá elevar la eficiencia en el proceso de revisión de los productos, brindando la posibilidad de dar respuesta a las solicitudes de los clientes en el menor tiempo posible.

- ✓ Se obtuvo como resultado la centralización del control estadístico de todas las métricas asignadas a un software en la Universidad de las Ciencias Informáticas.

- ✓ Con la definición del proceso anteriormente expuesto se garantizará una excelente gestión de las métricas de calidad del laboratorio de pruebas de software de CALISOFT.

Recomendaciones.

- ✓ Aplicar esta herramienta en otros laboratorios de pruebas de software con el objetivo de generalizar la propuesta para elevar la eficiencia de este proceso y evaluar las mejoras posibles.

- ✓ Diseñar a partir del sistema desarrollado y utilizando los datos que se almacenen, un sistema automatizado que permita la evaluación automática de los proyectos

Bibliografía.

- ✓ [Dra. Rodríguez Brisaboa, N, 2006] Red iberoamericana de tecnologías de software para la década del 2000 (ritos2). Disponible en: http://www.cyted.org. Fecha de acceso: 15 de enero 2011.

- ✓ [Condori-Fernández, N, Belenguer Faguás, J, Albiol, M.A, 2002] Modelo de Agregación Basado en un Sistema Neurodifuso para un Proceso de Evaluación de Calidad de Software. Disponible en: http://www.calidad.com.mx. Fecha de acceso: 15 de enero 2011.

- ✓ [Dra. Álvarez Cárdenas, S, Dra. Hernández González, A, Dra. Febles Estrada, A, Dr. Lau Fernández, R, 2006] Diplomado en Ingeniería del Software. Disponible en: http://dominoamericano.edu.do. Fecha de acceso: 15 de enero de 2011.

- ✓ [D.o.D. and U. Army, 2007] Official Practical Software and Systems Measurement Web Site. Disponible en: http://www.psmsc.com. Fecha de acceso: 15 de enero de 2011.

- ✓ [Fenton, N, 1997] Software quality assurance & Measurement. A worldwide perspective, Chapman&Hall. Disponible en: http://www.infor.uva.es. Fecha de acceso: 15 de enero de 2011.

- ✓ [Florac, W. A, Park, R. E, Carleton, A. D 1997] Practical Software Measurement: Measuring for Process Management and Improvement, Software Engineering Institute. Disponible en: http://mildpdf.com. Fecha de acceso: 15 de enero de 2011.

- ✓ [Humphrey, W. S 1995] A Discipline for Software Engineering. Disponible en: http://www.andrews.edu. Fecha de acceso: 15 de enero de 2011.

- ✓ [IEEE 1998] IEEE Std 1061-1998 IEEE Standard for a Software Quality Metrics Methodology. Fecha de acceso: 15 de enero de 2011.

- ✓ [ISO 2011] Official Site of International Organization for Standardization (ISO). Disponible en: http://www.iso.org. Fecha de acceso: 15 de enero de 2011.

- ✓ [Pressman, R. S 1998] Ingeniería de Software. Un Enfoque práctico.

- ✓ [ITEC Leonardo DaVinci, 2006] Gestión, Control y Garantía de la Calidad de Software. Disponible en: http://www.ldv.com.ar. Fecha de acceso: 16 de enero de 2011.

- ✓ [Fillottrani, P. R 2007] Calidad en el Desarrollo de Software. Modelos de calidad de software. Disponible en: http://www.cs.uns.edu.ar. Fecha de acceso: 16 de enero de 2011.

- ✓ [Joomla 2011] Sitio Oficial de Joomla. Disponible en: http://www.joomlaspanish.org. Fecha de acceso: 17 de enero de 2011.

- ✓ [Drupal 2011] Sitio Oficial de Drupal Hispano. Disponible en: http://drupal.org.es. Fecha de acceso: 17 de enero de 2011.

- ✓ [Visual Paradigm 2011] Official Site of Visual Paradigm. Disponible en: http://www.visual-paradigm.com. Fecha de acceso: 17 de enero de 2011.

- ✓ [PostgresSQL 2011] Official Site of PostgresSQL. Disponible en: http://www.postgresql.org. Fecha de acceso: 17 de enero de 2011.

- ✓ [Apache 2011] Official Site of Apache Software Foundation. Disponible en: http://www.apache.org. Fecha de acceso: 17 de enero de 2011.

- ✓ [González Doria, H 2001] Las Métricas de Software y su Uso en la Región. Disponible en: http://catarina.udlap.mx. Fecha de acceso: 15 de enero 2011.

- ✓ [NetBeans 2011] Official Site of NetBeans. Disponible en: http://netbeans.org. Fecha de acceso: 21 de enero 2011.

- ✓ [EVA. UCI 2011] Entorno Virtual de Aprendizaje. Disponible en: http://eva.uci.cu. Fecha de acceso: 9 de febrero 2011.

- ✓ [DAC-UCLA 2004] Portales Horizontales. Disponible en: http://www.ucla.edu.ve. Fecha de acceso: 16 de enero 2011.

- ✓ [IBM 2011] Official site of IBM. Disponible en: http://www.ibm.com. Fecha de acceso: 13 de febrero 2011.

- ✓ [IBM 2011] The analysis model. Disponible en: http://www.publib.boulder.ibm.com. Fecha de acceso: 26 de febrero 2011.
- ✓ [SQS 2011] Software Quality Systems S.A. Disponible en: http://www.sqs.es. Fecha de acceso: 13 de febrero 2011.
- ✓ [Cuzcano Quintín, Saúl 2011] Modelo de Análisis. Disponible en: http://scribd.com. Fecha de acceso: 26 de febrero 2011.
- ✓ [García Peñalvo, F 2008] Ingeniería del Software. Diseño orientado a objetos. Disponible en: http://scribd.com. Fecha de acceso: 26 de febrero 2011.
- ✓ [Tecnológico 2011] Diseño de la Arquitectura del Software. Disponible en: http://mitecnologico.com. Fecha de acceso: 10 de marzo 2011.
- ✓ [MAC 2011] Algunos tipos de Arquitectura. Disponible en: http://homepage.mac.com. Fecha de acceso: 10 de marzo 2011.
- ✓ [Carreño, J 2009] Arquitectura de Software. Disponible en: http://sophia.javeriana.edu.co. Fecha de acceso: 10 de marzo 2011.
- ✓ [Tutorial Drupal 2011] Arquitectura. Disponible en: http://www.cursosdrupal.com. Fecha de acceso: 10 de marzo 2011.
- ✓ [García Valón, E 2010] Herramienta para generar las estadísticas cuantitativas de las No Conformidades. Fecha de acceso: 20 de febrero 2011.
- ✓ [Visconti. M, Astudillo. H] Fundamentos de Ingeniería de Software. Disponible en: http://www.inf.utfm.cl. Fecha de acceso: 12 de marzo 2011.
- ✓ [Velasco Elizondo Perla Inés] Prueba de componentes de software basadas en el modelo de javabeans. Disponible en: http://www.cs.man.ac.uk/ ~velascop/publ/Tesis.pdf. Fecha de acceso: 15 de mayo 2011.
- ✓ Guía de técnicas de pruebas y métricas. Disponible en: http://www.lpsi.eui.upm.es/MDes/TfcMetrica/GTPrueb.htm. Fecha de acceso: 15 de mayo 2011.

Referencias Bibliográficas.

[i] Pressman, R.S: Ingeniería del Software. Un enfoque práctico. Mc Graw Hill, 2002.

[ii] ITEC Leonardo DaVinci, 2006: Gestión, Control y Garantía de la Calidad de Software.

[iii] Fillottrani, P. R 2007: Calidad en el Desarrollo de Software. Modelos de calidad de software.

[iv] Fillottrani, P. R 2007: Calidad en el Desarrollo de Software. Modelos de calidad de software.

[v] SQS 2011: Software Quality Systems S.A.

[vi] IBM 2011: Official site of IBM.

[vii] DAC-UCLA 2004. Portales Horizontales.

[viii] Eyssautier de la Mora, Maurice (2006).