QUALITYSOFT: IMPLEMENTACIÓN DE UN APLICATIVO INFORMÁTICO DE AUDITORÍA DE SISTEMAS BASADO EN LOS ESTÁNDARES INTERNACIONALES DE CALIDAD

CARLOS ALBERTO JOJOA CAICEDO
JUAN GUILLERMO PATIÑO DE LOS RIOS

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2011

QUALITYSOFT: IMPLEMENTACIÓN DE UN APLICATIVO INFORMÁTICO DE AUDITORÍA DE SISTEMAS BASADO EN LOS ESTÁNDARES INTERNACIONALES DE CALIDAD

CARLOS ALBERTO JOJOA CAICEDO
JUAN GUILLERMO PATIÑO DE LOS RIOS

Trabajo de Grado presentado como requisito parcial
para optar al título de Ingeniero de Sistemas

Director
Ing. Francisco Nicolás Javier Solarte Solarte

Codirector
Ing. Esp. José Javier Villalba Romero

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2011

NOTA DE ACEPTACION

_____
_____
_____
_____

_____
Firma del Presidente del Jurado

_____
Firma del Jurado

_____
Firma del Jurado

San Juan de Pasto, 14 de Febrero del 2011

“Las ideas y conclusiones aportadas en el Trabajo de Grado son responsabilidad exclusiva del autor.”

Artículo 1º del Acuerdo Nº. 324 de octubre 11 de 1996, emanado del Honorable Consejo Directivo de la Universidad de Nariño.

AGRADECIMIENTOS

Queremos agradecer a:

A Dios primero que todo quien es nuestro guía, quien nos dio la fuerza e inteligencia de sacar nuestra carrera y proyecto de tesis adelante, siempre estuvo con nosotros en los momentos más complicados y nunca nos desamparó, permitiéndonos lograr nuestras metas y objetivos.
A nuestros padres Guillermo Patiño y Luis Jojoa, a nuestras madres Aida De Los Ríos y Rosa Caicedo, quienes siempre estuvieron ahí en los momentos difíciles con una voz de consuelo, sufrieron con nuestros tropiezos y se alegraron de nuestros logros, quienes se preocuparon de formarnos como personas integrales en todo sentido, no existen palabras que permitan expresar la felicidad que sentimos de tener padres como ellos, llenos de amor y dedicación. Todo lo que somos se los debemos a ustedes.

A nuestros hermanos y hermanas, Iván Patiño y Adriana Jojoa, con los que compartimos tantas cosas, se convierten en nuestros mejores amigos, son personas incondicionales que nos brindan su apoyo y compresión, capaces de sacar una sonrisa en nuestros momentos de tristeza, capaces de darnos esperanzas en esos momentos de desesperación, capaces de darnos fortaleza en los momentos de debilidad.
A nuestros amigos y profesores personas con los que hemos contado siempre, son la fuente de conocimiento, compromiso y alegría, con los que hemos compartido muchos años de vida y muy buenos momentos.

A nuestros asesores Ingenieros de Sistemas Francisco Nicolás Javier Solarte Solarte y José Javier Villalba Romero, quienes fueron primero que todo nuestros amigos, fueron nuestra guía en este proyecto gracias a sus ideas, aportes y apoyo incondicional nació el proyecto QUALITYSOFT .

TABLA DE CONTENIDO

LISTA FIGURAS

LISTA TABLAS

LISTA ANEXOS

GLOSARIO

Aplicativo: Programa informático que permite a un usuario utilizar una computadora con un fin específico. Las aplicaciones son parte del software de una computadora, y suelen ejecutarse sobre el sistema operativo. [5]

Auditoria: Es una función de dirección cuya finalidad es analizar y apreciar, con vistas a las eventuales acciones correctivas, el control interno de las organizaciones para garantizar la integridad de su patrimonio, la veracidad de su información y el mantenimiento de la eficacia de sus sistemas de gestión. [3]

Automatización: Es el uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y/o procesos industriales sustituyendo a operadores humanos. [7]

Calidad: La calidad es una cualidad y propiedad inherente de las cosas, que permite que estas sean comparadas con otras de su misma especie. La definición de calidad nunca puede ser precisa, ya que se trata de una apreciación subjetiva. [1]

Calidad de Software: Características propias del software que se desea controlar y asegurar, el software es un producto inmaterial que no se fabrica, tampoco se degradan físicamente, sino que se desarrolla; El software puede tener errores, incidencias pero no son similares a lo que cualquier equipo de carácter físico. [7]

Ciclo de Vida: El ciclo de vida del software es la descripción de las distintas formas de desarrollo de un proyecto informático. [7]

Ciclo de Vida Ágil: Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. [7]

Componente Software: Son todos aquellos recursos desarrollados para un fin concreto y que puede formar solo o junto con otros, un entorno funcional requerido por cualquier proceso predefinido. [7]

Estándar: Base o modelo en el que todo el mundo se ha puesto de acuerdo. [7]

Gestión de Calidad de Software: Es un conjunto de actividades de la función general de la Dirección que determina la calidad, los objetivos y las responsabilidades. Se basa en la determinación y aplicación de las políticas de calidad de la empresa. [7]

IEEE: Corresponde a las siglas de (Institute of Electrical and ElectronicsEngineers) en español Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. [7]

Informático: conjunto de conocimientos científicos y técnicos que hacen posible el tratamiento automático de la información por medio de computadores. [6]

ISO: (Internacional Standards Organization) Organización Internacional de estándares fundada en 1946, con sede principal en Ginebra, ISO establece o fija estándares internacionales. Se ocupa de todos los campos, excepto la electricidad y la electrónica, que se rigen por la Internacional Electrotechnical Comisión (IEC), también en Ginebra. [7]

ISO 9001: Es un conjunto de normas sobre la calidad y la gestión. [7]

Métrica: Proporcionan una indicación de cómo se ajusta el software a los requisitos implícitos y explícitos del cliente. Es decir cómo voy a medir para que mi sistema se adapte a los requisitos que me pide el cliente. [4]

Modelo: Un modelo es una estructura conceptual que sugiere un marco de ideas para un conjunto de descripciones que de otra manera no podrían ser sistematizadas. [7]

Proceso de ingeniería de software: se define como un conjunto de etapas parcialmente ordenadas con la intención de logra un objetivo, en este caso, la obtención de un producto de software de calidad. [7]

Requerimientos: Características que se desea que posea un sistema o un software. [7]

Software: se conoce como software al equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos del sistema, llamados hardware.
Tales componentes lógicos incluyen, entre muchos otros, aplicaciones informáticas, como el procesador de textos, que permite al usuario realizar todas las tareas concernientes a la edición de textos o el software de sistema, tal como el sistema operativo, que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de las aplicaciones, proporcionando también una interfaz para el usuario [2]

RESUMEN

Este trabajo presenta el análisis, diseño e implementación de QUALITYSOFT, una herramienta de auditoría de sistemas para mejorar la evaluación de calidad de software.

En el presente proyecto se realizó una investigación de los estándares de calidad de software más conocidos para determinar cuál o cuáles de ellos pueden complementarse para posteriormente construir una herramienta de software que permita evaluar la calidad tomando en cuenta el estándar más completo.

QUALITYSOFT tiene como funcionalidad principal evaluar proyectos sobre calidad de desarrollo de software a través de características específicas, las cuales son seleccionadas por el evaluador, los indicadores dependerá de los objetivos de negocio para el producto y las necesidades del evaluador. Las necesidades son especificadas por los criterios de medidas.

QUALITYSOFT contiene tres herramientas principales:
El Módulo de Administración permite crear un nuevo proyecto y abrir proyectos realizados para mostrar reportes o borrar informes, también permite agregar y eliminar preguntas, modificar perfiles de usuario.

El módulo Crear proyecto permite crear un proyecto, ingresar datos del proyecto, seleccionar factores (indicadores) y evaluar las métricas.

El módulo Abrir proyecto permite visualizar todos los proyectos ya realizados, mostrar un informe sobre estos, además permite borrar proyectos y continuar desarrollando proyectos incompletos. También, permite hacer copias de seguridad de proyectos y restaurarlos.

En la última parte se muestran resultados de pruebas de funcionalidad de la herramienta de auditoría de sistemas con datos reales.

ABSTRACT

This paper presents the analysis, design and implementation of QUALITYSOFT, a systems audit tool to improve software quality assessment.

In this project we conducted an investigation of software quality standards known to determine which of them can be supplemented later to build a software tool for evaluating the quality taking into account the most comprehensive standard.

QUALITYSOFT main function is to evaluate projects on software development quality through specific characteristics, which are selected by the evaluator, the indicators will depend on business goals for the product and the needs of the evaluator. The requirements are specified by the criteria of measures.

QUALITYSOFT contains three main elements:
Management Module to create a new project and open reporting projects to display or delete reports also add and delete questions, change user profiles.

Create Project module lets you create a project, enter project data, select factors (indicators) and evaluation metrics.

Open Project module lets you view all completed projects, view the report on these in addition to delete projects and continue developing incomplete projects. Backup and restore projects.

In the last section presents results of testing functionality of the system auditing tool with real data.

INTRODUCCIÓN

La auditoría de sistemas de información se está convirtiendo en un factor crucial para la sociedad, debido a la gran dependencia que las organizaciones tienen de sistemas que gestionen su información, y debido también a la necesidad derivada de verificar la calidad de los servicios ofrecidos por estos sistemas de información, así como la de garantizar una adecuada seguridad que consista en una correcta confidencialidad, integridad y disponibilidad de los datos que gestionan y que son uno de los activos más importantes de las organizaciones.[8]

La calidad de desarrollo de software consiste en proporcionar la gestión para informar de los datos necesarios sobre la calidad del producto, por lo que se va adquiriendo una visión más profunda y segura de que la calidad del producto está cumpliendo sus objetivos. Es de esperar, que si los datos proporcionados mediante la garantía de la calidad identifican problemas, la gestión afronte los problemas y aplique los recursos necesarios para resolverlos.

Hoy en día debido a la creciente necesidad de realizar proyectos complejos en cortos periodos de tiempo, a la vez que con menores esfuerzos tanto humanos como económicos, está favoreciendo el avance de lo que se conoce como Calidad de Desarrollo de Software. Esta nueva disciplina se apoya en niveles de madurez software, que son combinados adecuadamente para satisfacer los requisitos del proyecto. [9]

Teniendo en cuenta lo anterior, se propuso este proyecto cuyo objetivo principal fue la implementación de un aplicativo informático que permita facilitar el proceso de auditoría de desarrollo de software basado en estándares internacionales de calidad, permitiendo a las empresas incrementar la productividad eficiente de producto software.

El proyecto se basó en la investigación de los estándares de calidad de software más conocidos para determinar cuál o cuáles de ellos pueden complementarse para posteriormente desarrollar construir una herramienta de software orientada a que permita evaluar la calidad tomando en cuenta el estándar más completo.

Con este aplicativo informático se da soporte automatizado para la realización de auditoría informática, y después, para verificar el funcionamiento correcto, eficaz y eficiente de la misma.

Para el estudio se tuvo en cuenta además el proceso de auditoría de sistemas para aplicar el proceso haciendo uso de la herramienta de software que permita evaluar de manera más objetiva la calidad del software.

El proyecto se llamó QUALITYSOFT: Implementación de un Aplicativo Informático de Auditoria de Sistemas Basado en los Estándares Internacionales de Calidad y se realizó bajo la modalidad de trabajo de investigación. El proyecto se desarrolló bajo la línea de Ingeniería de Software, aprobada por el Departamento de Sistemas mediante acuerdo No 045 del 10 octubre de 2002.

QUALITYSOFT fue el resultado de la investigación de estándares de calidad del software para determinar las métricas que permitan recoger todos los aspectos requeridos en la evaluación de software.

Se implementó un paquete informático que permitirá evaluar la calidad de desarrollo de Software mediante unos cuestionarios que serán las métricas basadas en los estándares seleccionados.

El paquete permite evaluar las características en su totalidad o de forma individual por cada uno de los ítems definidos dentro del estándar.

El producto informático se puede aplicar a cualquier tipo de software. No importando si es de tipo cliente servidor, distribuido o Web.

Como limitante el paquete sólo sirve para realizar la evaluación del software, pues la interpretación deberá ser dada por el auditor o persona que este evaluando el software.

Se implementó una interfaz gráfica amigable con el usuario que permite una interacción con el sistema y conteniendo una base de datos que permite actualizar y revisar los cuestionarios. Los reportes generados por el software ayudarán a hacer la interpretación del informe de auditoría.
Con la investigación de los estándares seleccionados, se hizo la implementación de una herramienta para la medición de calidad de software apoyando el proceso de auditoría de sistemas.

Con este proceso se garantiza la implementación de un producto más eficiente y la posibilidad de corregir errores en el tiempo de desarrollo para obtener al final un producto competitivo.

Además este producto informático de apoyo a la auditoria de sistemas orientado a la evaluación del software, puede ser la base de futuras investigaciones y herramientas que optimicen este proceso.

La descripción del problema que se planteó fue el siguiente:

El software que se está desarrollando en las empresas no cumple en su totalidad con los estándares de calidad [10], pues para la empresa significa el uso de mayores recursos de personal, económicos y tecnológicos.

La industria del software ha venido creciendo significativamente, en Colombia ha tenido un crecimiento del 12%, mientras en el mundo ha sido de solo el 7% [11]. Esto hace que cada vez más las empresas y las personas usen software para realizar sus tareas y funciones, por lo tanto se necesita tener calidad en el desarrollo del software.

La calidad del software es una preocupación a la que se dedican muchos esfuerzos. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios. La falta de calidad de software genera pérdidas en costo y tiempo a las empresas desarrolladoras debido a que el producto final puede presentar múltiples errores que repercuten en la ineficiencia de los procesos.

Los principales esfuerzos de la comunidad de software en estos temas se han basado hasta ahora en los aspectos funcionales de los componentes, es decir, en la funcionalidad que ofrecen. Sin embargo, por lo general se han venido obviando muchos de los aspectos de calidad que intervienen a la hora de seleccionar componentes y ensamblarlos para construir aplicaciones que satisfagan los requisitos del cliente. Este tipo de aspectos, que se llaman “extra-funcionales”, cada vez acapara más la atención de los arquitectos e ingenieros del software. [9]
Se puede considerar varias causas para esta omisión de la valoración en los requisitos extra-funcionales. La primera es que no existe una definición consensuada de las características o atributos de calidad que hay que medir. Así, se podrá encontrar diversas clasificaciones en la literatura: desde los factores de

calidad propuestos por McCall [McCall, 1977], el modelo de calidad de Boehm [Boehm et al., 1976], los estándares internacionales ISO-9126 [ISO, 1991] e ISO-14598 [ISO, 1998; ISO, 1999], la lista de atributos para la valoración citada para el modelo COCOTS [Abts et al., 2000] basada en estándares de IEEE (en particular el IEEE/ANSI 830-1993), y otras varias [Bosch 2000, Szyperski 1998].[9]

Añadido a todo esto, se puede encontrar una ausencia casi total de métricas que permitan dar una estimación más objetiva de estos atributos. Este hecho se ve afectado por la situación actual de los estándares internacionales relativos a la calidad del software como producto.

Por otro lado, es importante señalar que los estándares internacionales proporcionan guías y modelos de calidad para temas muy generales y su aplicación suele centrarse en temas de un gran espectro. Por tanto, no suelen estar pensadas para ofrecer soluciones en temas muy concretos.

Con base en lo anterior se formuló la descripción del problema así:

¿Cómo optimizar el proceso de Auditoria al software mediante la implementación de un paquete basado en la investigación de los estándares de la calidad de software?
Para la Sistematización del problema se tuvieron en cuenta los siguientes puntos:
¿Cómo definir las métricas de evaluación de desarrollo de software?
¿Cómo evaluar las métricas seleccionadas de los estándares?
¿Cómo generar reportes de la evaluación de calidad de software?
¿Cómo desarrollar una interfaz amigable con el usuario?
¿Cómo implementar una base de datos que soporte el paquete informático?
¿Cómo probar el paquete informático implementado?

Así mismo, para el desarrollo de QUALITYSOFT se tuvo en cuenta los siguientes objetivos:

Como objetivo general, “Optimizar el proceso de auditoría a la calidad del software mediante la implementación de un paquete informático basado en la investigación de los estándares de calidad de software”.
Y como objetivos específicos:

- Identificar los diferentes estándares de calidad de software aplicados para construcción y evaluación de software en diversas fuentes como Internet, libros, y otras fuentes.

- Analizar y seleccionar los estándares que permitan recoger todos los aspectos requeridos para la evaluación o construcción de software.
- Seleccionar las métricas más pertinentes de los estándares seleccionados basadas en criterios objetivos y establecer la escala de medición de las métricas.
- Desarrollar una herramienta informática que permita evaluar la calidad del software en el proceso de auditoría de sistemas, basado en la aplicación de las métricas seleccionadas.
- Implementar la herramienta informática que ayude a realizar el proceso de auditoría a la calidad del software y realizar las pruebas en los productos de software desarrollados en una de las instituciones de educación superior.

El proyecto se realizó bajo la siguiente justificación:

Para las empresas y las personas, la calidad de desarrollo de software es muy importante, pues en el mundo actual las mayorías de las industrias giran en torno a programas y aplicaciones que necesitan tener una buena calidad de desempeño para cumplir con los requisitos y funciones para los cuales fueron construidos.

En la actualidad es evidente que la información se ha convertido en uno de los activos principales de las organizaciones, representando en muchos casos su principal elemento estratégico para la realización de sus objetivos, y como soporte de su actividad. Las organizaciones invierten enormes cantidades de dinero y tiempo en la creación de sistemas de información que les ofrezcan la mayor productividad y calidad posible. [8]

La medición de la calidad de software en las empresas se ha convertido en un tema fundamental con el cual se pretende garantizar un buen comportamiento de sus productos en el ambiente para el cual fueron creados, pero el proceso de medición del la calidad se ha convertido en un proceso dispendioso el cual consume recursos y tiempo que muchos fabricantes de software prefieren no gastar.

En la actualidad se carece de aplicativos y paquetes informáticos que permitan realizar una adecuada y eficiente auditoria de sistemas para el desarrollo de software, lo cual conlleva a las empresas a omitir este proceso de auditoría y como consecuencia obtienen productos software sin la calidad necesaria para la realización de la actividad.

Con la implementación de este proyecto se pretende que las personas y las empresas cuenten con una herramienta informática que facilite y ayude tanto en el proceso de auditoría, como en la evaluación de la calidad de desarrollo del software, optimizando el consumo de recursos y tiempo.

Este aplicativo informático se utiliza para descubrir y detener los errores del software, y se lleva a cabo para evaluar eventos específicos, o bien para revisar todas las actividades de un sistema.

El control de la calidad permite realizar las rectificaciones necesarias a cualquier falla encontrada en el producto software.

La información obtenida mediante este aplicativo servirá para futuras comparaciones en las diferentes etapas de desarrollo del producto software y determinar si los errores encontrados fueron corregidos o no.

Para el desarrollo de este Proyecto se tuvieron en cuenta los siguientes antecedentes:

Para la evaluación de la calidad de software que utiliza como base las normas internacionales ISO, es fundamental tener métricas para la medición, de ahí la importancia de investigar resultados en este campo.

Uno de los primeros organismos que impulsaron el desarrollo de estándares en procesos de software fue el Departamento de Defensa de los Estados Unidos (DoD). A ellos les preocupaba el establecimiento de métricas para identificar a los contratistas potenciales referentes a ésta rama. De esta forma ellos crearon la Software Engineering Institute (SEI).

El SEI es un centro de investigación y desarrollo sostenido por el Departamento de Defensa y operado por la Universidad de Carnegie Mellon. El propósito principal de éste instituto es el ayudar a las empresas a buscar mejoras en cuanto a ingeniería de software basadas en métricas [SEI 03]. [24]

El primer mecanismo que sirvió para éste propósito fue un cuestionario para evaluar el estado actual sobre las prácticas de software en las empresas [Humphrey 87]. Gracias al éxito de éste, en 1988 la SEI creó un programa de entrenamiento a las organizaciones que querían realizar auto diagnósticos en su proceso de software. Éste programa de entrenamiento dio paso a la creación del

Software Process Assessment (SPA) cuyo objetivo esencial era hacer llegar el programa de entrenamiento a más empresas.

El mercado creado por la SEI fue aprovechado por particulares que empezaron a dar asesorías individuales por lo que éstos decidieron llevar a cabo un proyecto de grandes magnitudes que con el tiempo sería llamado Capability Maturity Model (CMM).

Como respuesta a la estandarización de procesos de software por parte de la SEI, ISO creó su apartado de calidad en cuanto al software. La International Organization for Standardization (ISO) fue la primera empresa encargada de crear un estándar en cuanto a creación de software se refiere. ISO es creada en 1947 al final de la segunda guerra mundial con la participación de delegados de 25 países con la finalidad de facilitar la coordinación internacional y la unificación de estándares industriales [ISO 2002b]. [24]

ISO 9001:1994 fue la respuesta a CMM por parte de ISO. Ésta se encarga de los "sistemas de calidad. Modelo para asegurar la calidad en diseño o desarrollo, producción, instalación y servicio". Tres años después salió al mercado el ISO 9000-3:1997 que es en realidad una expansión además de una guía para el ISO 9001:1994. [24]

ISO 9126 proviene desde el modelo establecido en 1977 por McCall y sus colegas, los cuales propusieron un modelo para especificar la calidad del software. Esta norma ISO 9126 distingue entre fallo y no conformidad. Un fallo es el incumplimiento de los requisitos previos, mientras que la no conformidad es el incumplimiento de los requisitos especificados. Una distinción similar es la que se establece entre validación y verificación. [25]

En Colombia existe la Federación Colombiana de la Industria del Software – FEDESOFT, la cual se creó en noviembre de 1999, con la misión de velar por el fortalecimiento del sector a través del desarrollo de políticas que normalizan, defienden y promueven los intereses de los industriales del software en Colombia. [26]

En el departamento de Nariño se encuentra un proyecto desarrollado en el CESMAG titulado "Diagnostico del estado de los sistemas de gestión de seguridad de la información (SGSI), con la aplicación de un software, en las instituciones de educación superior de San Juan de Pasto", hecho por Rafael Llerena Riascos y

José Daniel Guerra Erazo, que realiza la medición de la seguridad aplicando el estándar ISO 27001 basado en los niveles de madurez.

En este proyecto se evalúa la seguridad informática basado en una serie de cuestionarios tomados de una norma internacional, de este proyecto tomamos como base la forma de evaluación de los cuestionario para aplicar en nuestro proyecto y crear nuestra forma de medición de la calidad de software.

Prototipo mosca: Encontramos en Venezuela “Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software” desarrollado por: Luis E Mendoza, María A. Perez, Anna C Grimán. Para evaluar la calidad de los Sistemas, integrando el modelo de Calidad del Producto y el modelo de Calidad del Proceso de Desarrollo, soportado en los conceptos de la Calidad Total Sistémica.[27]
El cual sirvió para usar como modelo para el desarrollo de QUALITYSOFT, aunque no se tuvo acceso al aplicativo, se logro obtener un artículo donde se explica el funcionamiento de prototipo Mosca y se sacaron muchas ideas y se obtuvo una idea concreta de las funciones y estructura del aplicativo QUALITYSOFT.

1. MARCO TEORICO

1.1. AUDITORIA DE SISTEMAS

La auditoría en informática es la revisión y la evaluación de los controles, sistemas, procedimientos de informática; de los equipos de cómputo, su utilización, eficiencia y seguridad, de la organización que participan en el procesamiento de la información, a fin de que por medio del señalamiento de cursos alternativos se logre una utilización más eficiente y segura de la información que servirá para una adecuada toma de decisiones.
La auditoría en informática deberá comprender no sólo la evaluación de los equipos de cómputo, de un sistema o procedimiento específico, sino que además habrá de evaluar los sistemas de información en general desde sus entradas, procedimientos, controles, archivos, seguridad y obtención de información.
La auditoría en informática es de vital importancia para el buen desempeño de los sistemas de información, ya que proporciona los controles necesarios para que los sistemas sean confiables y con un buen nivel de seguridad. Además debe evaluar cableado estructurado, entorno de funcionamiento, sistema operativo, programas de aplicación, routers. [12]
Es importante este concepto de auditoría ya que nos permite guiar la investigación entorno al uso de herramientas informáticas, normas y metodologías de auditoría que son tenidas en cuenta para el desarrollo del trabajo.

1.2. CONCEPTOS BÁSICOS DE MÉTRICAS

La palabra métrica, es muy común asociarla con las palabras medición y medida, aunque estas tres son distintas. 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. La medida proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto. Métrica se define como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado o también una medida cuantitativa del grado en que el sistema, componente o proceso posee un atributo dado.
Un Indicador es la métrica o una combinación de métricas que proporcionan una visión profunda del proceso del software, del proyecto software o del producto en

sí. Un indicador proporciona una visión profunda que permite ajustar el producto, el proyecto o el proceso. El objetivo principal de los indicadores de proceso es evaluar las condiciones de funcionamiento de un proceso y poder tener una visión de la eficacia de un proceso existente. Durante un tiempo considerable se recopilan las métricas de todos los proyectos y se proporcionan indicadores para obtener mejoras en el software.

Los métodos de medición o cálculo son unas secuencias lógicas de operaciones y potenciales heurísticas, expresadas de forma genérica, que permite la realización de una descripción de actividad. El tipo de método de medición va a depender de la naturaleza de las operaciones utilizadas para cuantificar el atributo. Pueden distinguirse dos tipos: Subjetivo, cuando la cuantificación supone un juicio realizado por un ser humano, y Objetivo, cuando la cuantificación está basada en métodos numéricos.

Una escala es un conjunto de valores con propiedades definidas. Las escalas pueden ser escalas numéricas (Continua o Discreta) o escala Categórica. Los tipos de escala pueden ser: Nominal, Ordinal, Intervalo, entre otras.

En este sentido la métrica es la correspondencia de un dominio real o empírico a un mundo formal, matemático. La medida incluye al valor numérico o nominal asignado al atributo de un ente por medio de dicha correspondencia. Las métricas pueden ser de tipo directo cuando no dependen de ninguna métrica de otro atributo e indirecta cuando se deriva de una o más métricas de otros atributos.

Algunos ejemplos de Métricas Directas son: Longitud del Texto del Cuerpo de una Página (medido por cantidad de palabras), Cantidad de Enlaces Rotos Internos (medidos por la presencia de errores del tipo 404), Cantidad de Imágenes con Texto Alternativo (medido por la presencia de la etiqueta ALT o con texto no nulo, en cada una de las imágenes vinculadas a las páginas de un sitio Web).

Algunos ejemplos de Métricas Indirectas son: Porcentaje de Enlaces Rotos de un Sitio, Porcentaje de Presencia de la propiedad ALT.

Ahora se estudiará brevemente el amplio campo de las métricas del software, ya que es la técnica que junto a revisiones, pruebas y gestión de configuración constituye el conjunto principal de medios operativos para el aseguramiento de la calidad en el proyecto.

En general, para la evaluación de la calidad es más habitual centrarse en medidas del producto que en medidas de procesos, aunque estas últimas pueden ser útiles para medir ciertos aspectos como la fiabilidad, se mide el tiempo medio entre fallos de software a lo largo de algún período de prueba, etc. Para mejorar cualquier proceso se debe medir atributos del proceso, definir y desarrollar un juego de métricas para esos atributos, y utilizar las métricas para encontrar indicadores para la estrategia de mejora. [7]

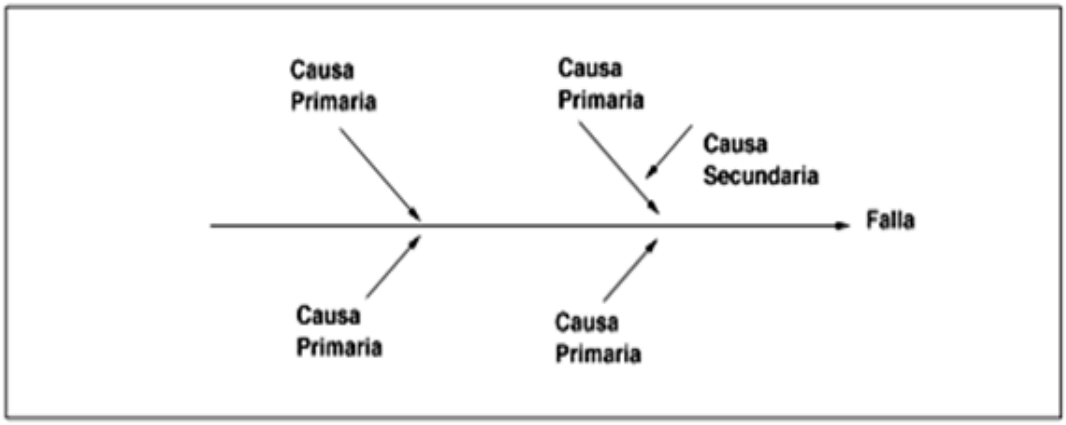
La eficacia de un proceso de software se mide a través de un juego de métricas según los resultados que provienen del proceso. Dentro de estos resultados se debe incluir: Medida de los errores detectados antes de la entrega del software, defectos detectados, productos de trabajo entregados, esfuerzo humano y tiempo consumido y ajuste con la planificación.

También se debe incluir métricas para medir las características de tareas específicas de la ingeniería del software, como la medida del tiempo y del esfuerzo para llevar a cabo actividades de protección, actividades genéricas de ingeniería de software.

Para una organización es importante estar a gusto con la recopilación y la utilización de métricas de proceso, de éstas se deriva la identificación de indicadores llevando a un enfoque más riguroso denominado Mejora estadística del proceso de software (MEPS). Este enfoque utiliza el análisis de fallas del software para recopilar información de errores y defectos. Para realizar un análisis de fallas se debe seguir los siguientes pasos: Categorizar por origen de todos los errores y defectos, Registrar el costo de corregir cada error o defecto, Contar el número de errores y defectos de cada categoría, calcular el costo global de errores y defectos de cada categoría, para posteriormente, desarrollar planes para eliminar los errores y defectos más costosos.

Para determinar las principales causas que pueden ocasionar anomalías en el software y con base en ello extraer los indicadores que permitan a una organización de software modificar su proceso para reducir la frecuencia de errores y defectos se utiliza el diagrama de espina. En el diagrama de espina, la línea central, representa el factor de calidad o el problema en consideración. Las líneas diagonales conectadas a la línea central indican una causa potencial del problema de calidad. [7]

Figura 1. Diagrama de espina de pescado: Análisis de problemas y causas de calidad del software (tomado del módulo evaluación de software)



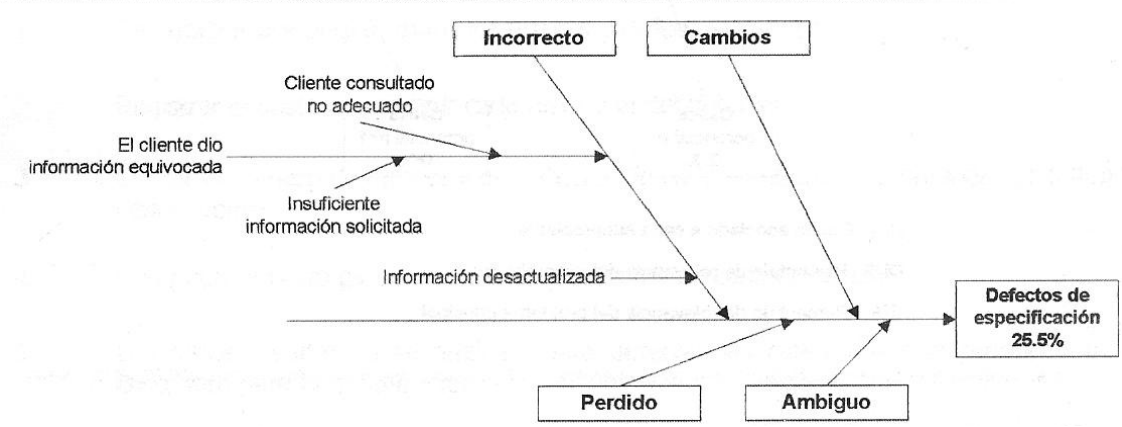
Por ejemplo, en la tabla 1 se muestra las causas y su origen de fallas en un proyecto de software:

Tabla 1. Origen de errores o defectos y causas en un proyecto software

Origen de errores o defectos	**Causa**	**%**
Especificación de requisitos	Lógica	20
	Manejo de datos	10.9
	Estándares	6.9
Diseño	Especificaciones	25.5
Código	Interfaz de software	6.0
	Interfaz de Hardware	7.7
	Comprobación de errores	10.9
	Interfaz de usuario	11.7

En la tabla anterior se muestra el origen de errores o defectos y causas en un proyecto software tomado del módulo evaluación de software
Utilizando el diagrama de espina para la identificación de las causas específicas para este problema, sería:

Figura 2. Identificación de causas de errores o defectos en un software (tomado del módulo evaluación de software)



Las métricas del Proyecto se utilizan para minimizar la planificación de desarrollo, ya que realizan ajustes y minimizan los retrasos. Además se utilizan para evaluar la calidad de los productos. A medida que mejora la calidad se minimizan los defectos. Las métricas del proyecto de software sugiere que los proyectos deben medir: las entradas, la dimensión de los recursos que se requieren para realizar el trabajo, las salidas, medidas de las entradas o productos creados durante el proceso de ingeniería de software, y resultados; medidas que indican la efectividad de las entregas. [7]

1.3. MÉTRICAS DEL SOFTWARE

Las métricas del software se pueden categorizar en:

1.3.1. Medidas directas: Dentro de estas se pueden incluir: el costo y el esfuerzo aplicado, las líneas de código producidas (LCD), la velocidad de ejecución, el tamaño de la memoria y los defectos informados durante un periodo de tiempo establecido. [7]

1.3.2. Métricas indirectas: Dentro de estas están la funcionalidad, la calidad, la complejidad, la eficiencia, la confiabilidad, La facilidad de uso y la facilidad de mantenimiento.

1.3.3. Métricas orientadas al tamaño: Provienen de la normalización de las medidas de calidad y/o productividad considerando el tamaño del software que se haya producido.

1.3.4. Métricas orientadas a la función: Las métricas del software orientadas a la función permiten la medida de la funcionalidad de la aplicación. Esta métrica busca identificar los factores críticos que determinan el tamaño del software y por consiguiente, estimar el esfuerzo y el costo para desarrollarlo. De aquí nace la técnica de análisis de puntos de función, que mide una aplicación con base en las funciones que éste realiza por solicitud del usuario final. [7]

Los puntos de función se obtienen utilizando una función empírica basado en medidas cuantitativas del dominio de información del software y valoraciones subjetivas de la complejidad del software. Los puntos de función se calculan utilizando la siguiente tabla:

Tabla 2. Tabla de cálculo de puntos de función

Parámetros de medición	**Valor Obtenido**	**Simple**	**Medio**	**Complejo**
Número de entradas de usuario		0-3	4-5	6
Número de salidas de usuario		0-4	5-6	7
Número de peticiones de usuario		0-3	4-5	6
Número de archivos		0-7	10-14	15
Número de interfaces externas		0-5	7-9	10

La tabla anterior representa un ejemplo de la evaluación con puntos de función en donde las empresas deben llenar los valores obtenidos por cada campo y asignar rangos de acuerdo a las normas y estándares propios.

Se determinan 5 características del ámbito de la información y los cálculos aparecen en la posición apropiada en la tabla. Los valores del ámbito de información están definidos de la siguiente forma:

Tabla 3. Características y definición de puntos de función (a)

Característica	**Definición**
Número de entradas de usuario	Se cuenta cada entrada de usuario que proporcione al software diferentes datos orientados a la aplicación
Número de salidas de usuario	Se cuenta cada salida que proporciona al usuario información orientada a la aplicación. En este contexto las salidas se refieren a informes, pantallas, mensajes de error
Número de peticiones de usuario	Una petición está definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva. Se cuenta cada petición por separado
Número de archivos	Se cuenta cada archivo maestro lógico
Número de interfaces externas	Se cuentan todas las interfaces legibles por la máquina por ejemplo: archivos de datos, en cinta o discos que son utilizados para transmitir información a otro sistema.

Cuando han sido recogidos los datos anteriores, se asocia el valor de complejidad a cada cuenta. Las organizaciones que utilizan métodos de puntos de función desarrollan criterios para determinar si una entrada es denominada simple, media o compleja. No obstante la determinación de la complejidad es algo subjetivo.

$$PF = Cuenta\_total * [0.65 + 0.01* \text{sumatoria } (f_i)]$$

De la fórmula anterior, el significado de los valores constantes es:

0.01: Este valor es usado para convertir el valor de la sumatoria (f_i) a porcentaje lo que es los mismo que 1/100.

0.65: Este valor es tomado como ejemplo para la explicación de la formula aunque el valor puede variar entre el rango 0<valor<1. El valor es puesto por la empresa dependiendo de sus criterios de evaluación.

Tabla 4. Características y definición de puntos de función (b)

<table>
<tr><td>PF</td><td>Punto de función</td></tr>
<tr><td>Cuenta_total</td><td>Es la suma de todas las entradas obtenidas</td></tr>
<tr><td>f_i</td><td>Donde i=1 hasta 14. son los valores de ajuste de la complejidad basados en las respuestas a las cuestiones señaladas de la siguiente tabla:

Evaluar cada factor de 0 a 5

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>No influencia</td><td>Incidental</td><td>Moderado</td><td>Medio</td><td>Significativo</td><td>Esencial</td></tr>
</table>

<table>
<tr><td>F_i :</td><td></td></tr>
<tr><td>1</td><td>¿Requiere el sistema copias de seguridad y de recuperación fiable?</td></tr>
<tr><td>2</td><td>¿Se requiere comunicación de datos?</td></tr>
<tr><td>3</td><td>¿Existen funciones de procesamiento distribuido?</td></tr>
<tr><td>4</td><td>¿Es crítico el rendimiento?</td></tr>
<tr><td>5</td><td>¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?</td></tr>
<tr><td>6</td><td>¿Requiere el sistema entrada de datos interactivo?</td></tr>
<tr><td>7</td><td>¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?</td></tr>
<tr><td>8</td><td>¿Se actualizan los archivos maestros de forma interactiva?</td></tr>
<tr><td>9</td><td>¿Son complejas las entradas, las salidas, los archivos o las peticiones?</td></tr>
<tr><td>10</td><td>¿Es complejo el procesamiento interno?</td></tr>
</table>
</td></tr>
</table>

11	¿Se ha diseñado el código para ser reutilizable?
12	¿Están incluidas en el diseño la conversión y la instalación?
13	¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?
14	¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?

Una vez calculados los puntos de función se usan como medida de la productividad, calidad y otros productos del software.

Productividad= PF/persona-mes
Calidad= Errores/ PF
Costo= Pesos/PF
Documentación= Páginas_documentadas/ PF

1.4. DEFINICIÓN DE CALIDAD

Inicialmente la calidad hace referencia al proceso industrial donde W. E. Deming propuso la idea de calidad como "conformidad a requisitos y confiabilidad en el funcionamiento". Posteriormente surgen otras definiciones de calidad como la de J. Juran que propone una definición breve: “Quality is fitness for use”; es decir, "la calidad es la adecuación del producto al uso" donde esta definición incluye las características del producto que permiten obtener la satisfacción del usuario y, además, supone la ausencia de deficiencias. Para P. Crossby el concepto de Juran se asume pero también destaca la prevención: “Cero defectos” o “Hacerlo bien a la primera”. En la bibliografía son frecuentes las definiciones de calidad pero en su gran mayoría todas ellas se acercan al concepto expresado por Juran.
La definición oficial, y más completa, es la de la norma ISO 9000:2000 que define la calidad en general como: “Grado en el que un conjunto de características inherentes cumple con los requisitos”, donde los requisitos son las necesidades o expectativas establecidas, generalmente implícitas u obligatorias y las características se refieren a cualquier tipo de rasgo diferenciador. También se debe recordar que las necesidades pueden variar en el tiempo, por lo que hay que prever la revisión de la especificación. [7]

Esta definición permite comprender que la consecución de la calidad puede tener tres orígenes:

1.4.1. La calidad realizada: Que es la calidad obtenida por la persona que realiza el trabajo gracias a su habilidad en la ejecución de una tarea. Se potencia con la mejora de las habilidades personales y técnicas de los participantes en un proceso determinado. [7]

1.4.2. La calidad programada: Que es la calidad que se ha encomendado conseguir a la persona responsable de ejecutar el trabajo. Se potencia con la elaboración de una especificación que sirva de buena referencia a los participantes en un proceso. Esta aparece descrita en una especificación, en un documento de diseño o en un plano constructivo. [7]

1.4.3. La calidad necesaria: La que el cliente exige con mayor o menor grado de concreción o, al menos, la que le gustaría recibir. Se potencia con una adecuada obtención de información de la idea de calidad de los clientes y de su percepción de la misma. [7]

La gestión de la calidad pretende entonces, conseguir que estos tres objetivos coincidan entre sí.

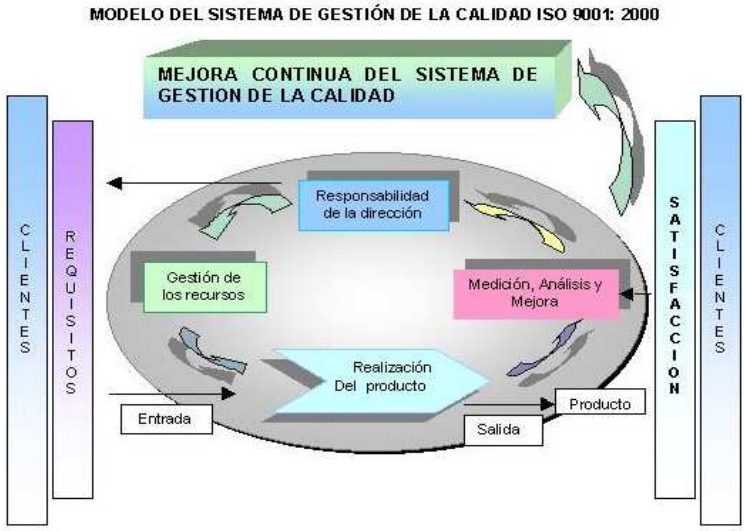
1.5. NORMATIVIDAD DE CALIDAD

La normativa de calidad surge inicialmente en empresas de algunos de los sectores de seguridad crítica como el militar, nuclear, y aeroespacial. La normativa nuclear ha inspirado muchas otras normas, sin embargo, la revolución normativa para todos los sectores se produjo con la implementación de la serie de normas ISO 9000, inicialmente con la norma de gestión y aseguramiento de la calidad ISO 9000, y posteriormente tres normas con recomendaciones para el aseguramiento externo de la calidad, la ISO 9001, 9002 y 9003 dependiendo de qué parte del ciclo de la calidad fuera el centro de actividad de la empresa y una norma ISO 9004 con recomendaciones para la gestión interna de la calidad.

Sin embargo, la reciente revisión de la serie 9000 realizada en el año 2000, cambió la filosofía y la estructura de las normas 9000, ahora existe una única norma ISO 9001:2000 que abarca todos los aspectos necesarios de todas las actividades del ciclo de vida. Las organizaciones que antes empleaban las normas ISO 9002 y 9003, aplicarán la ISO 9001:2000 limitando su ámbito de aplicación. El

cambio de filosofía experimentado se basa en incrementar la importancia de la mejora continua y el ciclo PDCA (*Plan-Do-Check-Act*), así como una mayor definición de los procesos y de la evaluación de la satisfacción de clientes y usuarios. La importancia de estas normas reside en que se emplean como base para que las empresas se certifiquen en procesos y transmitan a sus clientes la confianza en que trabaja con procedimientos que permiten asegurar la calidad en sus actividades. [7]

Figura 3. Modelo de gestión de calidad ISO 9001: 2000 (tomado del módulo evaluación de software)



En el caso del software que es un producto muy especial, fue necesario hacer una interpretación de la versión ISO 9001:1994 generando la guía ISO 9000-3:1997 que es la “Guía para aplicar ISO 9001 al desarrollo, suministro y mantenimiento de software”. Con la llegada de la norma ISO 9001:2000, la guía ISO 9000-3 es aplicable y útil.

La guía contempla tres aspectos principales de disposiciones adaptadas a la terminología y características especiales del software como producto, el primero es el marco de trabajo de la empresa (sistema de calidad, responsabilidad de la dirección y la realización de acciones correctivas), el segundo que muestra las actividades del ciclo de vida (Revisión de contrato, especificación, planificación, planificación de la calidad, diseño, implementación, revisiones, pruebas, aceptación, replicación, entrega, instalación y mantenimiento), y el tercero donde están las actividades de apoyo (Gestión de configuración, control de documentos,

métricas, convenciones y reglas, herramientas, formación, registros de calidad y compras). [7]

Desde el punto de vista práctico, la ISO 9001:2000 incluye las siguientes disposiciones:

1.5.1. Responsabilidad de la dirección: Desde un ámbito general muestra el compromiso con la satisfacción del cliente, promoviendo una determinación eficaz de requisitos y de las necesidades del cliente, establece una política de calidad que debe llevar a una planificación de la calidad y de sus objetivos, a través de un sistema de gestión de calidad en el que deben existir revisiones formales de la gestión realizada. [7]

1.5.2. Gestión de recursos: Para determinar y proporcionar lo necesario para la gestión de calidad, especialmente en el ámbito de los recursos humanos donde debe realizarse la adecuada política de asignación a tareas, determinar, realizar y evaluar el impacto de las acciones de formación y cualificación, y de competencias profesionales. No se deben olvidar otros recursos como la información, las infraestructuras necesarias y el entorno de trabajo. [7]

1.5.3. Gestión de los procesos: Con disposiciones para las actividades de gestión, los procesos relacionados con el cliente, el diseño y el desarrollo de productos y servicios, la gestión de compras y proveedores, la producción y la operación de servicios, el control de las no-conformidades de las entregas respecto de los requisitos establecidos y los servicios post-entrega.

1.5.4. Medición, análisis y mejora: Contempla la medición del rendimiento del sistema, medición de procesos, medición de productos y/o servicios y el control de la propia medición y de los medios de inspección y prueba. En cuanto a análisis de datos, se trata de obtener las conclusiones apropiadas para emprender acciones de mejora relacionadas con corrección de errores, prevención de problemas para el futuro y el establecimiento de procesos de mejora continua. [7]

1.6. INGENIERÍA DE SOFTWARE Y CALIDAD

Para hablar de la calidad en el software, se debe tener en cuenta que este es un producto con características particulares, por lo cual se hace necesario adaptar la terminología creada y aplicada en los sectores industriales. Algunas de estas características del software son las siguientes:
El software se desarrolla, no se fabrica ya que todo el costo de producción se centra en el diseño de la primera copia. El software es un producto lógico, el verdadero producto del software es el diseño de una serie de instrucciones para el computador, su existencia en papel o en soporte magnético no es más que una representación de las instrucciones en un código o lenguaje.

El software no se degrada con el uso, ya que la naturaleza lógica del software permite que permanezca inalterable por muy intensa que sea su utilización. Se puede degradar su representación magnética pero no su esencia.

La complejidad del software, la ausencia de controles adecuados hace que el software sea entregado muchas veces y conscientemente con defectos, incluso públicamente declarados. En el sector informático, incluso, se llega a cobrar una cuota de mantenimiento para reparar los defectos que el propio productor del software ha entregado con el mismo.

Un porcentaje muy grande de la producción de software se hace aún a la medida. En vez de emplear componentes existentes y ensamblarlos, aunque las bibliotecas de funciones o componentes están ya cambiando en parte esta situación.

El software es extraordinariamente flexible, ya que se puede cambiar con facilidad reutilizando trozos de un producto para construir otro. Sin embargo, la facilidad para cambiarlo es también un peligro que hay que controlar.

El diccionario estándar de ingeniería del software IEEE Std.6l0, indica que el software son “los programas de ordenador, los procedimientos y, posiblemente, la documentación asociada y los datos relativos a la operación del sistema informático”. No se limita al código solamente y se debe tener en cuenta el software en cualquier estado de evolución (diseños, especificaciones, datos de prueba, etc.), si se quiere obtener calidad en el software. También conviene recordar que la calidad del software se debe obtener a medida que se construye; no es un añadido que se pueda poner una vez desarrollado el software. [7]

1.7. GESTIÓN DE LA CALIDAD DEL SOFTWARE

La definición del estándar IEEE Std.610-1991, indica que calidad del software es el grado con el que un sistema, componente o proceso cumple con los requisitos especificados y las necesidades expectativas del cliente o usuario. Esta definición es la que más se ajusta al concepto de concordancia del software producido con los requisitos explícitamente establecidos, con los estándares de desarrollo expresamente fijados y con los requisitos implícitos, no establecidos formalmente, que desea el usuario.

Los requisitos se reflejan en la especificación de requisitos de manera explícita, el documento constituye la culminación de la etapa de análisis dentro del proceso de desarrollo. Los requisitos pueden ser funcionales, cuando se determinan las funciones que debe realizar el software y no funcionales como el rendimiento, la seguridad, el tiempo de respuesta, la interfaz, etc. De igual forma, los estándares y las normas, determinan cómo se debe realizar el proceso de desarrollo de software. Además, existen requisitos implícitos, no expresamente declarados, pero que el usuario del software desea obtener.

Para hacer el estudio de la calidad del software se debe conocer primero los principales términos empleados en esta área, algunos de ellos son:

1.7.1. Gestión de la calidad del software (*Software Quality Management*):Son las actividades coordinadas para dirigir y controlar una organización en lo relativo a la calidad. Esto se puede entender como el aspecto de la función general de la gestión que determina y aplica la política de calidad (objetivos y directrices generales de calidad de una empresa). Normalmente la gestión de calidad se aplica a nivel de empresa, por lo que incluye planificación estratégica, asignación de recursos, etc. [7]

1.7.2. Aseguramiento de la calidad del software (*Software Quality Assurance*):Es la parte de la gestión de la calidad orientada a proporcionar confianza en que se cumplirán los requisitos de la calidad. A nivel del software, podría presentarse como el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto satisfará los requisitos dados de calidad. También puede referirse, en el software al “conjunto de actividades para evaluar el proceso mediante el cual se desarrolla el producto”. El aseguramiento pretende dar confianza en que el producto tiene calidad. [7]

1.7.3. Control de la calidad del software (*Software Quality Control*):Es la parte de la gestión de la calidad orientada al cumplimiento de los requisitos de la calidad. Suele incluir las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales, el de mantener bajo control un proceso y eliminar las causas de defectos en las diferentes fases del ciclo de vida. En general, son las actividades para evaluar la calidad de los productos desarrollados. También, en el software, puede ser el "proceso de verificar el propio trabajo o el de un compañero". [7]

1.7.4. Verificación y validación del software (*Software Verification and Validation*): Que es una actividad ligada al control de la calidad en el ámbito del software. La verificación hace referencia a comprobar si los productos desarrollados en una fase del ciclo de vida satisfacen los requisitos establecidos en la fase anterior. Se suele decidir si el producto está completo y es consistente. Aquí se realizan las actividades para comprobar si un producto software está técnicamente bien ensamblado y si funciona. Y la validación que se refiere a comprobar si el software desarrollado satisface los requisitos del usuario. Por lo tanto, son las actividades de comprobación de que el producto de software desarrollado es el que se deseaba, es decir, si el producto funciona como el usuario quiere y realiza las funciones que se habían solicitado. [7]

1.8. ORGANIZACIÓN INTERNACIONAL PARA LA ESTANDARIZACIÓN O ISO:

Es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

La ISO es una red de los institutos de normas nacionales de 163 países, sobre la base de un miembro por país, con una Secretaría Central en Ginebra (Suiza) que coordina el sistema. La Organización Internacional de Normalización (ISO), con sede en Ginebra, está compuesta por delegaciones gubernamentales y no gubernamentales subdivididos en una serie de subcomités encargados de desarrollar las guías que contribuirán al mejoramiento ambiental.

Las normas desarrolladas por ISO son voluntarias, comprendiendo que ISO es un organismo no gubernamental y no depende de ningún otro organismo internacional, por lo tanto, no tiene autoridad para imponer sus normas a ningún país.

Está compuesta por representantes de los organismos de normalización (ON) nacionales, que produce normas internacionales industriales y comerciales. Dichas normas se conocen como normas ISO y su finalidad es la coordinación de las normas nacionales, en consonancia con el Acta Final de la Organización Mundial del Comercio, con el propósito de facilitar el comercio, el intercambio de información y contribuir con normas comunes al desarrollo y a la transferencia de tecnologías.[13]

1.9. NORMA ISO/IEC 9126

Las computadoras están siendo utilizadas en una variedad cada vez mayor de áreas de aplicación, y su correcta operación es a menudo crucial para el éxito del negocio y/o seguridad de las personas. El desarrollo o selección de altos productos de calidad de software, es de primordial importancia. Integrar especificación y evaluación de la calidad del producto de software es un factor clave para garantizar una calidad adecuada. Esto puede ser logrado mediante la definición de las características de calidad adecuados, teniendo en cuenta la finalidad del uso del producto de software. Es importante que todos los productos de referencia de software tengan característica de calidad especificada y evaluada, siempre que sea posible utilizando métricas validadas o aceptadas. ISO / IEC 9126 (1991): la evaluación de productos de software - Características de calidad y directrices para su uso, que se desarrolló para apoyar a estas necesidades, se definen seis características de calidad y se describe un software de evaluación de productos modelo de proceso. [7]

Como características de calidad y métricas asociadas puede ser útil no sólo para la evaluación de un producto software, sino también para definir los requisitos de calidad y la utilización de otros, la norma ISO / IEC 9126 (1991) ha sido sustituido por dos normas relacionadas con varias partes: ISO / IEC 9126 (calidad de los productos software) e ISO / IEC 14598 (evaluación del producto de software). Las características de calidad del producto software definidos en esta parte de la norma ISO / IEC 9126 se pueden utilizar para especificar los clientes funcionales y no funcionales y los requisitos del usuario.

Esta parte de la norma ISO / IEC 9126 es una revisión de la norma ISO / IEC 9126 (1991), y conserva las mismas características de calidad del software. Las principales diferencias son:

• La introducción de subcaracterísticas normativas, la mayoría de los cuales se basan en los informativos de subcaracterísticas en la norma ISO / IEC 9126 (1991).
• La especificación de un modelo de calidad.
• La introducción de la calidad en uso.
• Eliminación del proceso de evaluación (que ahora se especifica en la norma estándar ISO / IEC 14598).
• Coordinación del contenido de la norma ISO / IEC 14598-1. [7]

1.10. NORMA DE EVALUACIÓN ISO/IEC 9126

Esta norma Internacional fue publicada en 1992, la cual es usada para la evaluación de la calidad de software, llamado “Information technology-Software product evaluation-Quality characteristics and guidelines for their use”; o también conocido como ISO 9126 (o ISO/IEC 9126). Este estándar describe 6 características generales: Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad, y Portabilidad.

La norma ISO/IEC 9126 permite especificar y evaluar la calidad del software desde diferentes criterios asociados con adquisición, requerimientos, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y Auditoría de software. Los modelos de calidad para el software se describen así: [7]

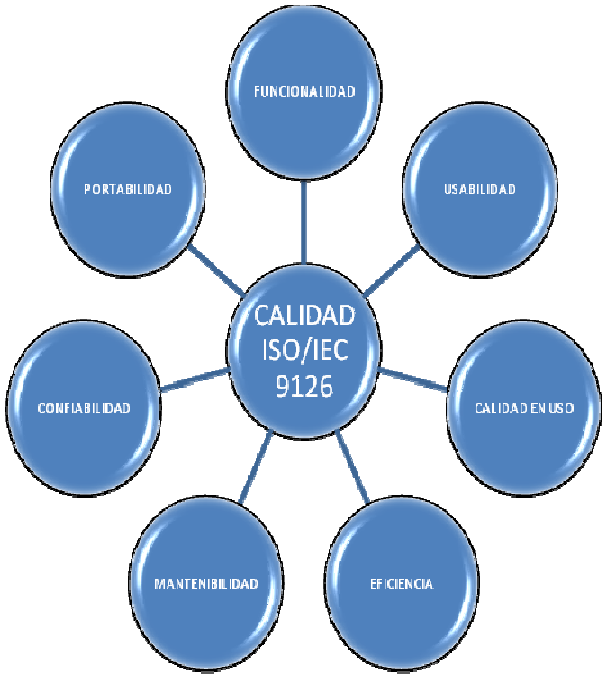
Calidad interna y externa: Especifica 6 características para calidad interna y externa, las cuales, están subdivididas. Estas divisiones se manifiestan externamente cuando el software es usado como parte de un sistema Informático, y son el resultado de atributos internos de software.

Calidad en uso: Calidad en uso es el efecto combinado para el usuario final de las 6 características de la calidad interna y externa del software. Especifica 4 características para la calidad en uso.

Al unir la calidad interna y externa con la calidad en uso se define un modelo de evaluación más completo, se puede pensar que la usabilidad del modelo de

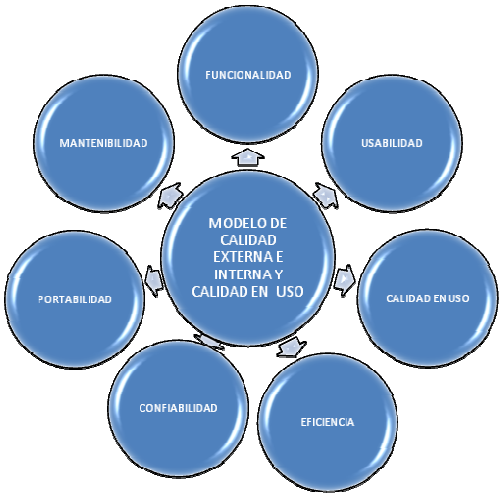
calidad externa e interna pueda ser igual al modelo de calidad en uso, pero no, la usabilidad es la forma como los profesionales interpretan o asimilan la funcionalidad del software y la calidad en uso se puede asumir como la forma que lo asimila o maneja el usuario final. Si se unen los dos modelos, se puede definir que los seis indicadores del primer modelo tienen sus atributos y el modelo de calidad en uso sus 4 indicadores pasarían hacer sus atributos, [7] mirándolo gráficamente queda así:

Figura 4. Norma de evaluación ISO/IEC 9126



Se establecen categorías para las cualidades de la calidad externa e interna y calidad en uso del software, teniendo en cuenta estos 7 indicadores (Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad y Calidad en uso), que se subdividen a su vez en varios indicadores; estas se pueden medir por métrica interna o externa. [7]

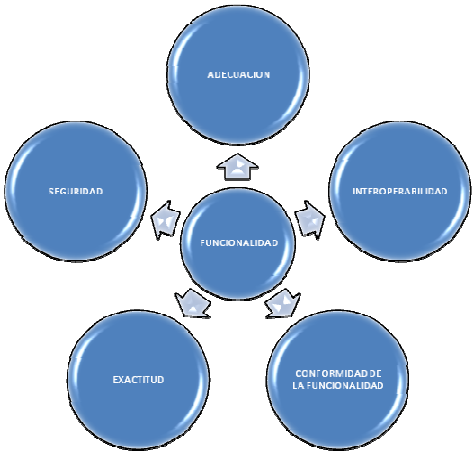
Figura 5. Evaluación interna, externa y calidad en uso ISO/IEC 9126



Las definiciones se dan para cada característica y subcaracterística de calidad del software que influye en la calidad. Para cada característica y subcaracterística, la capacidad del software es determinada por un conjunto de atributos internos que pueden ser medidos. Las características y sub características se pueden medir externamente por la capacidad del sistema que contiene el software.

1.10.1. Funcionalidad: Funcionalidad es la capacidad del software de cumplir y proveer las funciones para satisfacer las necesidades explícitas e implícitas cuando es utilizado en condiciones específicas. A continuación se muestra la característica de Funcionalidad y las subcaracterísticas que cubre:

Figura 6. Característica de funcionalidad



Esta figura representa la Característica Funcionalidad con sus sub_características asociadas

La Funcionalidad se divide en 5 criterios:

Adecuación: Es la capacidad del software para proveer un adecuado conjunto de funciones que cumplan las tareas y objetivos especificados por el usuario.

Exactitud: Es la capacidad del software para hacer procesos y entregar los resultados solicitados con precisión o de forma esperada.

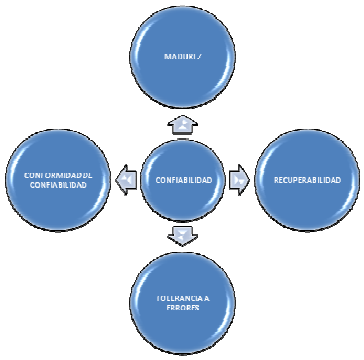
Interoperabilidad: Es la capacidad del software de interactuar con uno o más sistemas específicos.

Seguridad: Es la capacidad del software para proteger la información y los datos de manera que a los usuarios o los sistemas no autorizados se les niegue el acceso a ellos para realizar operaciones, y la capacidad de aceptar el acceso a los datos de los usuarios o sistemas autorizados

Conformidad de la funcionalidad: Es la capacidad del software de cumplir los estándares referentes a la funcionalidad.

1.10.2 Confiabilidad: La confiabilidad es la capacidad del software para asegurar un nivel de funcionamiento adecuado cuando es utilizando en condiciones especificas. En este caso la confiabilidad se amplía para sostener un nivel especificado de funcionamiento y no una función requerida.

Figura 7. Característica de confiabilidad



Esta figura representa la Característica Confiabilidad con sus sub_características asociadas

La Confiabilidad se divide en 4 criterios:

Madurez: Es la capacidad que tiene el software para evitar fallas cuando encuentra errores. Ejemplo, la forma como el software advierte al usuario cuando realiza operaciones en la unidad de disquete vacía, o cuando no encuentra espacio suficiente en el disco duro donde se está almacenando los datos.

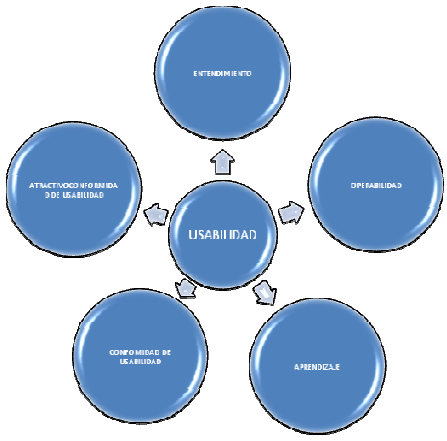
Tolerancia a errores: Es la capacidad que tiene el software para mantener un nivel de funcionamiento en caso de errores.

Recuperabilidad: Es la capacidad que tiene el software para restablecer su funcionamiento adecuado y recuperar los datos afectados en el caso de una falla.

Conformidad de la confiabilidad: Es la capacidad del software de cumplir a los estándares o normas relacionadas a la fiabilidad.

1.10.3. Usabilidad: La usabilidad es la capacidad del software de ser entendido, aprendido, y usado en forma fácil y atractiva. Algunos criterios de funcionalidad, fiabilidad y eficiencia afectan la usabilidad, pero para los propósitos de la ISO/IEC 9126 ellos no clasifican como usabilidad. La usabilidad está determinada por los usuarios finales y los usuarios indirectos del software, dirigidos a todos los ambientes, a la preparación del uso y el resultado obtenido. [7]

Figura 8. Característica de usabilidad



Esta figura representa la Característica Usabilidad con sus sub_características asociadas

La Usabilidad se divide en 5 criterios:

Entendimiento: Es la capacidad que tiene el software para permitir al usuario entender si es adecuado, y de una manera fácil como ser utilizado para las tareas y las condiciones particulares de la aplicación. En este criterio se debe tener en cuenta la documentación y las ayudas que el software entrega.

Aprendizaje: Es la forma como el software permite al usuario aprender su uso. También es importante considerar la documentación.

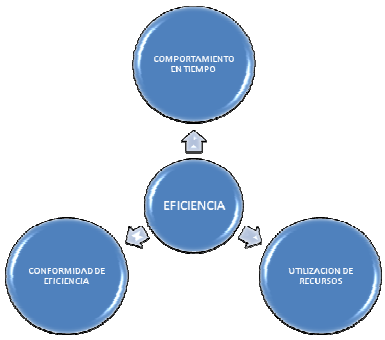
Operabilidad: Es la manera como el software permite al usuario operarlo y controlarlo.

Atractivo conformidad de usabilidad: Es la presentación del software que debe ser atractiva al usuario. Esto se refiere a las cualidades del software para hacer más agradable al usuario, ejemplo, el diseño gráfico.

Conformidad de usabilidad: Es la capacidad del software para cumplir los estándares o normas relacionadas a su usabilidad.

1.10.4. Eficiencia: La eficiencia del software es la forma del desempeño adecuado, de acuerdo al número de recursos utilizados según las condiciones planteadas. Se debe tener en cuenta aspectos como la configuración de hardware, el sistema operativo, operatividad de las redes, integración entre sistemas operativos entre otros.

Figura 9. Característica de eficiencia



Esta figura representa la Característica Eficiencia con sus sub_características asociadas

La Eficiencia se divide en 3 criterios:

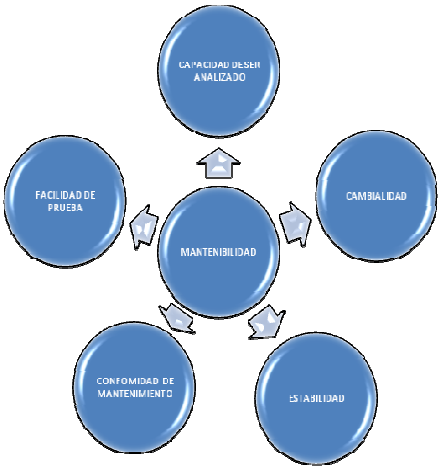
Comportamiento en tiempos: Son los tiempos adecuados de respuesta y procesamiento, el rendimiento cuando realiza su función en condiciones especificas. Ejemplo, ejecutar el procedimiento más complejo del software y esperar su tiempo de respuesta, realizar la misma función pero con más cantidad de registros.

Utilización de recursos: Es la capacidad del software para utilizar cantidades y tipos adecuados de recursos cuando este funciona bajo requerimientos o condiciones establecidas. Ejemplo, los recursos humanos, el hardware, dispositivos externos.

Conformidad de eficiencia: Es la capacidad que tiene el software para cumplir con los estándares o convenciones relacionados a la eficiencia.

1.10.5 Mantenibilidad: El mantenimiento es la cualidad que tiene el software para ser modificado. Incluyendo correcciones o mejoras del software, a cambios en el entorno, y especificaciones de requerimientos funcionales.

Figura 10. Característica de mantenibilidad



Esta figura representa la Característica de Mantenibilidad con sus sub_características asociadas

La Mantenibilidad se divide en 5 criterios:

Capacidad de ser analizado: Es la forma como el software permite diagnósticos de deficiencias o causas de fallas, o la identificación de partes modificadas.

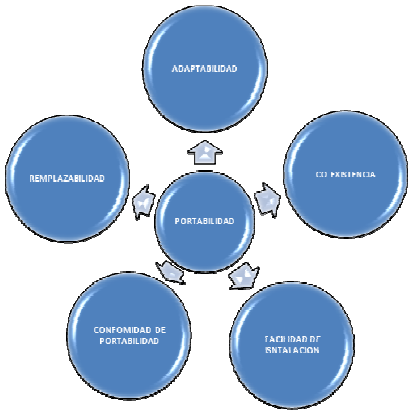
Cambiabilidad: Es la capacidad del software para que la implementación de una modificación se pueda realizar, incluye también codificación, diseño y documentación de cambios.

Estabilidad: Es la forma como el software evita efectos inesperados para modificaciones del mismo.
Facilidad de prueba: Es la forma como el software permite realizar pruebas a las modificaciones sin poner el riesgo los datos.

Conformidad de mantenimiento: Es la capacidad que tiene el software para cumplir con los estándares de facilidad de mantenimiento.

1.10.6. Portabilidad: Es la capacidad que tiene el software para ser trasladado de un entorno a otro.

Figura 11. Característica de portabilidad



Esta figura representa la Característica Portabilidad con sus sub_características asociadas

La Portabilidad se divide en 5 criterios:

Adaptabilidad: Es como el software se adapta a diferentes entornos especificados (hardware o sistemas operativos) sin que implique reacciones negativas ante el cambio. Incluye la escalabilidad de capacidad interna (Ejemplo: Campos en pantalla, tablas, volúmenes de transacciones, formatos de reporte, etc.).

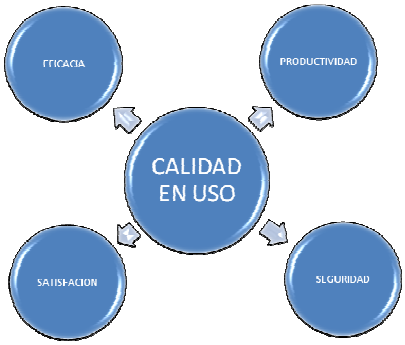
Facilidad de instalación: Es la facilidad del software para ser instalado en un entorno especifico o por el usuario final.

Coexistencia: Es la capacidad que tiene el software para coexistir con otro o productos de software, tales como compartir recursos comunes o dispositivos.
Remplazabilidad: Es la capacidad que tiene el software para ser remplazado por otro del mismo tipo, y para el mismo objetivo. Ejemplo, la remplazabilidad de una nueva versión es importante para el usuario o la propiedad de poder migrar los datos a otro software de diferente proveedor.

Conformidad de portabilidad: Es la capacidad que tiene el software para cumplir con los estándares relacionados a la portabilidad. [7]

1.10.7. Calidad en uso: Es la calidad del software que el usuario final refleja, la forma como el usuario final logra realizar los procesos con satisfacción, eficiencia y exactitud. La calidad en uso debe asegurar la prueba o revisión de todas las opciones que el usuario trabaja diariamente y los procesos que realiza esporádicamente relacionados con el mismo software. [7]

Figura 12. Característica de calidad en uso



La Calidad en Uso se divide en 4 criterios:

Eficacia: Es la capacidad del software para permitir a los usuarios finales realizar los procesos con exactitud e integridad.

Productividad: Es la forma como el software permite a los usuarios emplear cantidades apropiadas de recursos, en relación a la eficacia lograda en un contexto específico de uso. Para una empresa es muy importante que el software no afecte la productividad del empleado

Seguridad: Se refiere a que el Software no tenga niveles de riesgo para causar daño a las personas, instituciones, software, propiedad intelectual o entorno. Los riesgos son normalmente el resultado de deficiencias en la funcionalidad (Incluyendo seguridad), fiabilidad, usabilidad o facilidad de mantenimiento.

Satisfacción: Es la respuesta del usuario a la interacción con el software, e incluye las actitudes hacia el uso del mismo. [7]

1.10.8. Campo de aplicación: Esta norma ISO / IEC 9126 describe un modelo de dos partes para la calidad del producto de software, la calidad interna y externa. Parten del modelo específico seis características para calidad interna y externa, que se subdividen en subcaracterísticas. Estas subcaracterísticas se manifiestan externamente cuando el software se utiliza como parte de un equipo de sistema, y son el resultado de los atributos del software interno. Este sub ítem de la norma ISO / IEC 9126 no entra en detalles del modelo de calidad interna y externa por debajo del nivel de subcaracterísticas.

Las características definidas son aplicables a todo tipo de software, incluyendo programas de computadoras y los datos contenidos en el firmware. Las características y subcaracterísticas proporcionan una terminología coherente para la calidad de productos de software. También proporcionan un marco para la especificación de requisitos de calidad para software, y hacer un equilibrio entre las capacidades del producto de software.

La norma ISO/IEC 9126 proporciona recomendaciones y requisitos para las métricas de producto de software. Estos indicadores son aplicables al especificar los requisitos de calidad y los objetivos de diseño para productos de software, incluidos los productos intermedios. Una explicación de cómo este modelo de

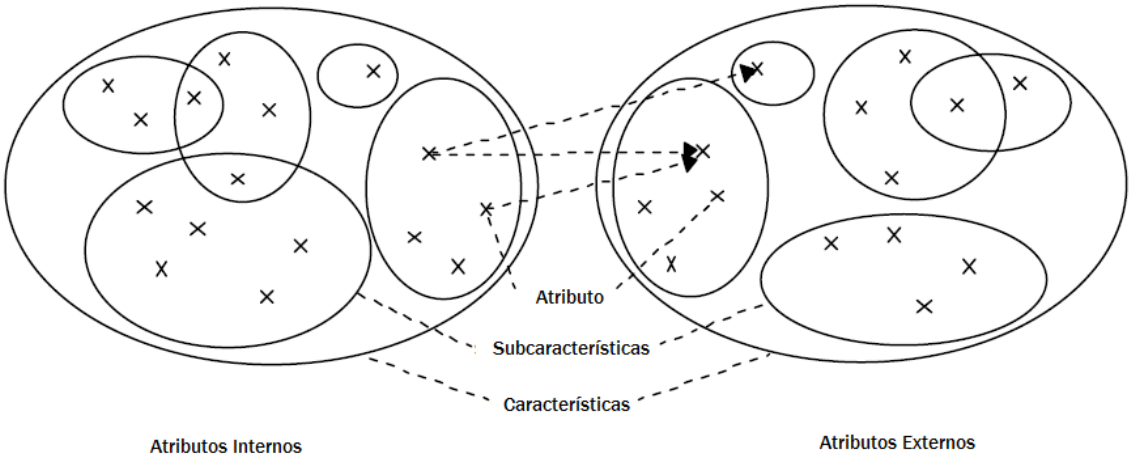
calidad se pueden aplicar en evaluación de productos de software se incluye en la norma ISO / IEC 14598-1.

Además la norma ISO / IEC 9126 permite evaluar la calidad de producto de software según las especificaciones de las diferentes perspectivas de los asociados con la adquisición, requerimientos, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y la auditoría de software. Puede ser utilizado por ejemplo por los promotores, adquirientes, personal de aseguramiento de la calidad y los evaluadores independientes, en particular los responsables de la especificación y evaluación de la calidad del producto de software. Ejemplos de usos de la calidad modelo definido en esta parte de la norma ISO / IEC 9126 son los siguientes:

• validar la integridad de una definición de los requisitos.
• Identificar los requisitos de software.
• identificar los objetivos de diseño de software.
• identificar los objetivos de pruebas de software.
• identificar los criterios de garantía de calidad.
• Identificar los criterios de aceptación de un producto de software completo. [14]

1.10.9. Métricas de software: Atributos Internos y externos. Los niveles de ciertos atributos internos se han encontrado para influir en los niveles de algunos atributos externos, por lo que no es tanto un aspecto externo y un aspecto interno a la mayoría de características. Por ejemplo, la confiabilidad se puede medir observando el exterior el número de fallos en un período determinado de tiempo de ejecución durante una prueba del software, e internamente mediante la inspección de las especificaciones detalladas y de código fuente para evaluar el nivel de tolerancia a fallos. Los atributos internos se dice que son indicadores de los atributos externos. Un atributo interno puede influir en una o más características, y una característica puede estar influenciada por más de un atributo (Figura 13). En este modelo la totalidad de software de los atributos de calidad de productos se clasifican en una estructura de árbol jerárquico de las características y subcaracterísticas. El nivel más alto de esta estructura se compone de las características de calidad y el más bajo nivel consiste en los atributos de calidad de software. La jerarquía no es perfecta, ya que algunos atributos pueden contribuir a más de una subcaracterística.

Figura 13. Atributos (tomado de la norma ISO/IEC 9126-1)



Las subcaracterísticas pueden ser medidas por métricas internas o por parámetros externos.

La correlación entre los atributos internos y las medidas externas nunca es perfecta, y el efecto que un determinado atributo interno en una medida externa asociada será determinado por la experiencia, y dependerá del contexto particular en que se utilice el software.

De la misma manera, las propiedades externas (tales como idoneidad, precisión, tolerancia a fallos o comportamiento en el tiempo) influirán en la calidad observada. Una falla en la calidad en uso (por ejemplo, el usuario no puede completar la tarea) se puede remontar a los atributos externos de calidad (por ejemplo, la idoneidad o la operabilidad) y el interno asociados a los atributos que tienen que ser cambiado. [14]

1.10.10. Métricas internas: Las métricas internas pueden ser aplicadas a un producto de software no ejecutable (como una especificación o código fuente) durante el diseño y la codificación. Al desarrollar un producto software intermedio, los productos deben ser evaluados utilizando métricas internas que miden las propiedades intrínsecas, como los que se pueden derivar de un comportamiento simulado. El propósito principal de estos indicadores internos es garantizar que la calidad requerida externa y la calidad en uso se consigue: ejemplos se dan en ISO / IEC 9126-3.

Las métricas internas proporcionan a los usuarios, los evaluadores, los probadores y desarrolladores beneficios que permiten de evaluar la calidad de productos de software y abordar los problemas de calidad antes que el producto de software se convierta en ejecutable.

Las métricas internas miden atributos internos o indican los atributos externos mediante el análisis de las propiedades de los productos de software intermedio o entrega. Las mediciones de parámetros internos, números de uso o las frecuencias de los de elementos de la composición de software que aparecen por ejemplo en la fuente instrucciones de código, el gráfico de control de flujo de datos y las representaciones del estado de transición. [14]

1.10.11. Métricas externas: Las métricas externas usan medidas de un producto software derivadas de las medidas del comportamiento del sistema del que forma parte, mediante pruebas, de funcionamiento y observando el software ejecutable o del sistema.

Antes de la adquisición o el uso de un producto de software, deben ser evaluados mediante indicadores basados en los objetivos de negocios relacionados con el uso, explotación y gestión del producto en una determinada organización y técnicas del medio ambiente. Estos son los principales indicadores externos: ejemplos se dan en la norma ISO / IEC 9126-2.

Las métricas en general, proporcionan a los usuarios externos, los evaluadores, los probadores y desarrolladores beneficios que permiten evaluar la calidad del producto de software durante las pruebas o la operación. [14]

1.11. CALIDAD DEL PRODUCTO SOFTWARE

1.11.1 NORMA ISO/IEC 9126: La norma ISO/IEC 9126 presentan dos modelos de calidad, el primero referido a la calidad interna y externa y el segundo modelo referido a la calidad en uso. A continuación se describe cada uno de ellos.

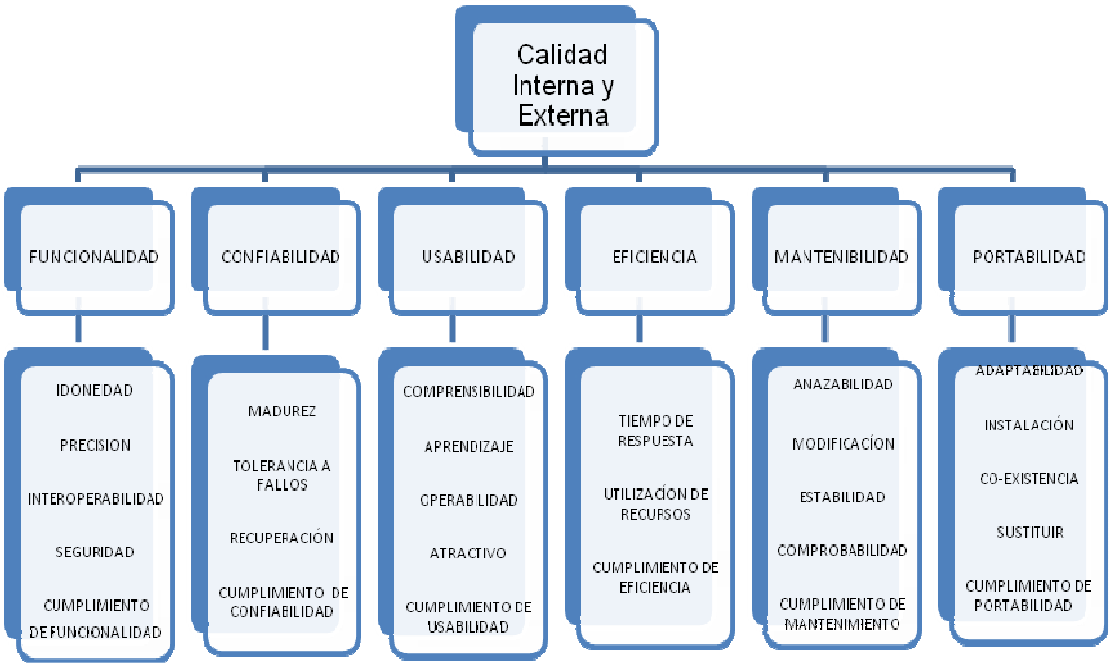
Calidad interna y externa

La norma ISO/IEC 9126 define la calidad interna como la totalidad de las características del producto software desde una perspectiva interna. La calidad interna es medida y evaluada en base a los requerimientos de calidad interna.

La calidad externa se define como la totalidad de las características del producto software desde una perspectiva externa. Mira la calidad del software cuando es ejecutado, la cual es típicamente medida y evaluada mientras se prueba en un ambiente simulado, con datos simulados y usando métricas externas. Durante las pruebas, muchas fallas serán descubiertas y eliminadas. Sin embargo algunas fallas todavía pueden permanecer después de las pruebas. Como es difícil corregir la arquitectura de software u otros aspectos fundamentales del diseño del software, el diseño base permanece sin cambios a través de las pruebas.

En la siguiente figura se representa el modelo de calidad interna o externa, se muestra un conjunto de seis características: Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad.

Figura 14. Modelo de calidad interna y externa del producto software



En la siguiente tabla se muestra las seis características y las definiciones de cada una de ellas.

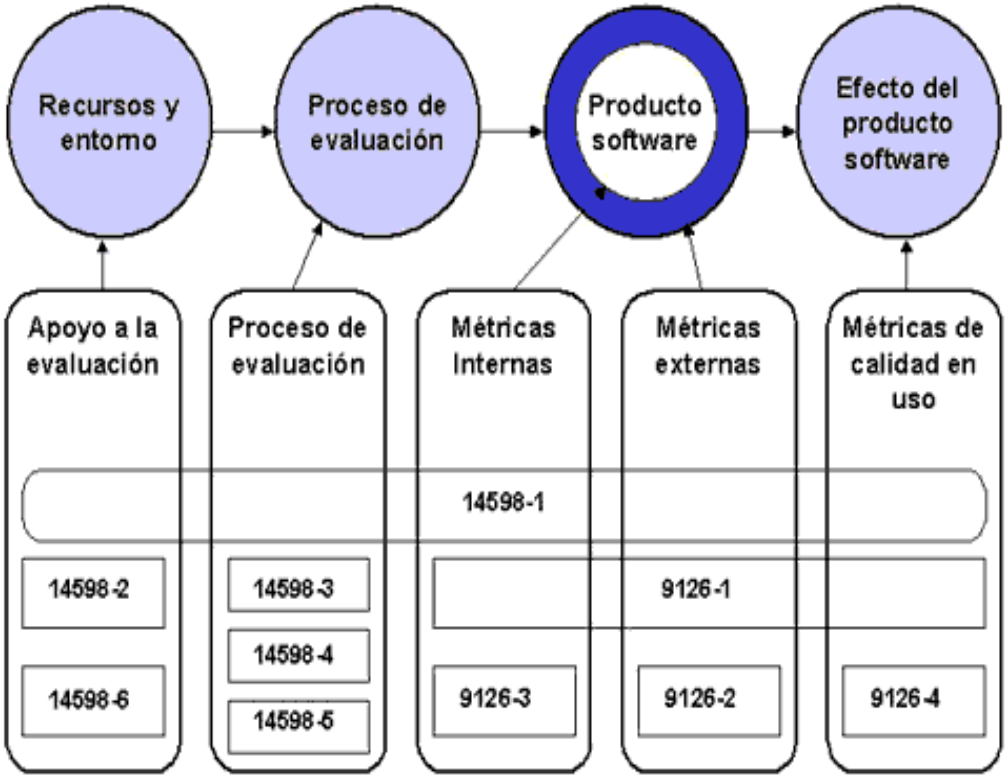
Tabla 5. Características de la calidad interna y externa ISO/IEC 9126.

Característica	**Definición**
Funcionalidad	Capacidad del producto software para proveer las funciones que satisfacen las necesidades explícitas e implícitas cuando el software se utiliza bajo condiciones específicas.
Confiabilidad	Capacidad del producto software para mantener un nivel especificado de funcionamiento cuando se está utilizando bajo condiciones especificadas
Usabilidad	Capacidad del producto software de ser entendido, aprendido usado y atractivo al usuario, cuando es usado bajo las condiciones especificadas
Eficiencia	Capacidad del producto software para proveer un desempeño apropiado, de acuerdo a la cantidad de recursos utilizados y bajo las condiciones planteadas
Mantenibilidad	Capacidad del producto software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, en requerimientos y especificaciones funcionales
Portabilidad	Capacidad del producto software de ser trasladado de un entorno a otro

1.11.2. NORMA ISO/IEC 14598: El estándar ISO/IEC 14598 es actualmente usado como base metodológica para la evaluación del producto software. los productos de software son solo una parte de la historia, también es necesario considerar mediciones en el proceso empleado para diseñar, desarrollar, probar, y controlar el producto. en esto juega un papel importante la norma en referencia, dado que ofrece una visión general, explica la relación entre su serie y el modelo de calidad de la iso/iec 9126, define los términos técnicos utilizados, contiene requisitos generales para la especificación y evaluación de la calidad del software, y clarifica los conceptos generales. además provee un marco de trabajo para evaluar la calidad de todos los tipos de productos software y establece requisitos para los métodos de medición y evaluación de los productos de software.

Es importante señalar que, la serie de normas ISO/IEC 14598 proporcionan un marco de trabajo para evaluar la calidad de todos los tipos de productos de software e indica los requisitos para los métodos de medición y para el proceso de evaluación. La norma ISO/IEC 14598 consta de seis partes que describen los requisitos del proceso de evaluación en tres situaciones diferentes: Requisitos para desarrolladores (parte 3), Requisitos para compradores (parte 4), Requisitos para evaluadores (parte 5).

Figura 15. Norma ISO/IEC 14598 (tomado del módulo evaluación de software)



1.11.2.1. Parte 1: Visión general: Básicamente provee una visión general de las otras cinco partes y explica la relación entre la evaluación del producto software y el modelo de calidad definido en la ISO/IEC 9126. Adicionalmente, hace la presentación del proceso de evaluación desglosado en los siguientes pasos: Establecer los requerimientos de evaluación, especificar la evaluación, planear la evaluación, Ejecutar la evaluación.

1.11.2.2. Parte 2: Planificación y gestión: Esta parte contiene los requerimientos y las guías para las funciones de soporte tales como el planeamiento y gestión para la evaluación del producto del software. Fundamentalmente, en esta parte, se planifican las mediciones y las actividades de evaluación, específicamente se incluye: Preparación de las políticas, definición de objetivos organizacionales y de mejora, identificación de la tecnología, asignación de responsabilidades, Identificación e implementación de técnicas de evaluación para software desarrollado y adquirido, entrenamiento en tecnología, recopilación de datos y herramientas, comparación y administración de mejoras dentro de la organización.

1.11.2.3. Parte 3: El Proceso para desarrolladores: Esta parte provee los requerimientos y las recomendaciones para la evaluación del producto software cuando la evaluación es conducida en paralelo con el desarrollo y llevada a cabo por el desarrollador. Se enfoca en el uso de indicadores que pueden predecir la calidad final del producto midiendo los productos intermedios que se desarrollan durante el ciclo de vida. Esta parte cubre el planeamiento y evaluación de mediciones internas y externas con el fin de asegurar que la calidad del producto sea incorporada en la fase de desarrollo. Una vez identificadas las características fundamentales de calidad y el marco de trabajo, deben ser definidas las etapas siguientes:

Organización: Los aspectos organizacionales de desarrollo y de soporte deben formar parte de todo el sistema de calidad y del plan de mediciones.

Planeamiento del Proyecto y requerimientos de Calidad: El desarrollo y el ciclo de vida de soporte deben ser establecidos y documentados durante el plan de calidad o en otros documentos. Es de vital importancia verificar que el productor y las medidas de control requeridas sean técnicamente factibles, razonables y alcanzables.

Especificaciones: En esta fase, el desarrollador realiza un mapeo de los requerimientos internos y externos de calidad, con relación a las especificaciones. Los requerimientos de mediciones resultantes de esta fase deben ser un tipo de mapeo entre las especificaciones de requerimientos, requerimientos externos de calidad, requerimientos internos correspondientes de calidad y atributos especificados junto a sus escalas de medición y valores objetivos que contribuyan a la cuantificación de la calidad del software, todo esto puede enfocarse por proyecto o por producto.

Diseño y planeamiento: Los procedimientos requeridos para el análisis y recopilación de datos necesitan ser definidos. De esta manera, el plan incluirá: Cronogramas, asignación de responsabilidades, uso de herramientas, bases de datos y entrenamiento especializado requerido. La precisión de las mediciones y técnicas estadísticas deben ser especificadas. En esta fase también deberá considerarse como los resultados de las mediciones impactarán en el desarrollo y los planes de contingencia y de mejora.

Montaje y pruebas: durante la etapa de montaje y pruebas, las mediciones actuales son recolectadas, se realizan análisis apropiados y se toman acciones necesarias. En cada fase del desarrollo debe procurarse lograr un montaje primeramente enfocado a las características internas y externas de calidad que definan la calidad global del producto y que puedan ser validadas por los resultados de las pruebas y la experiencia del usuario.

Y como etapa final del proyecto, deberá ser conducida una revisión general para determinar la efectividad global del ejercicio de recolección, establecer la validez de las métricas usadas e identificar puntos en los cuales podrían obtenerse beneficios para proyectos futuros. El resultado de esta revisión podría retroalimentar directamente el lanzamiento de futuros productos.

1.11.2.4. Parte 4: El proceso para compradores: Esta parte provee los requerimientos y las recomendaciones para que la evaluación del producto software sea conducida en función a los compradores que planean adquirir o re-usar un producto de software existente o pre-desarrollado. Los que adquieren el producto pueden comprar paquetes completos ya sea desarrollados según ciertas especificaciones o pre-desarrollados para un mercado más general. Los compradores también podrían ser desarrolladores que desean integrar productos estándar en sus propios diseños de software, o de desarrolladores buscando herramientas específicas de software. Al respecto, son necesarias cuatro etapas:

Establecimiento de los requerimientos: El alcance de la evaluación necesita ser establecido. Los requerimientos para la calidad del software definidos en la ISO/IEC 9126 pueden ser usados como punto de partida pero otros aspectos como el costo y deberán ser también considerados el de cumplimiento a regulaciones. El tiempo de la evaluación necesita ser consistente con los objetivos; enfoques muy tempranos podrían no proporcionar una figura adecuada

de la situación mientras que enfoques muy tardíos podrían ser muy limitados en su uso.

Especificación de la Evaluación: Durante la redacción de las especificaciones, debe considerarse: Los requerimientos de calidad a ser evaluados, correlacionados con la calidad en uso y métricas externas con prioridades, además de un umbral de aceptación definido, el alcance y lo que cubren los casos de prueba donde sean aplicables referencias a módulos de evaluación, los métodos de recolección de mediciones, información requerida y métodos de análisis.

Diseño de la Evaluación: El tipo de evaluación depende del tipo de software que está siendo evaluado. Software bajo desarrollo puede ser abordado en puntos discretos durante el desarrollo o cuando esté completo. Un plan de evaluación necesita considerar: Necesidades de acceso a la documentación del producto, herramientas de desarrollo y personal, requerimientos en costos y conocimientos, cronograma de evaluación y arreglos de contingencia, y criterio para decisiones de evaluación, métodos y herramientas de reporte, procedimientos para la validación y estandarización sobre proyectos futuros.

Ejecución de la Evaluación: Aunque esta etapa podría ser simplemente un registro en un libro de seguimiento, podría tenerse la necesidad de incluir: Los resultados mismos y la trazabilidad del producto así como información de configuración, registros de análisis, resultados y decisiones, problemas, limitaciones en las mediciones y cualquier compromiso con relación a los objetivos originales, conclusiones sobre los resultados de la evaluación pero también sobre los métodos empleados.

1.11.2.5. Parte 5: El Proceso para evaluadores: Esta parte provee los requerimientos y recomendaciones para la evaluación del producto software cuando la evaluación es conducida por evaluadores independientes. En esta parte tiene un rol importante los requerimientos de evaluación, las especificaciones de evaluación, el diseño de la evaluación, las actividades de evaluación y el reporte de evaluación. Estas etapas son resumidas a continuación:

Requerimientos de Evaluación: Los requerimientos deberían definir adicionalmente: La extensión del la cobertura (o el alcance), los objetivos de

evaluación y métodos de reporte, y las calificaciones e independencia requeridas de un evaluador.

Especificación de la Evaluación: Las especificaciones deberían cubrir adicionalmente: Definición del alcance y formato en las métricas empleadas identificando como deberán ser derivadas a partir de los requerimientos del producto, la identificación de mediciones no determinanticas para asegurar que ciertos niveles de frecuencia y objetividad requeridos sean obtenidos, y la identificación de métodos de correlación con relación a los resultados de las mediciones.

Se tienen identificadas tres sub-actividades con relación a la especificación de la evaluación: El análisis de la descripción del producto, la especificación de las mediciones a ser realizadas, y la verificación de la especificación resultante frente a los requerimientos de evaluación.

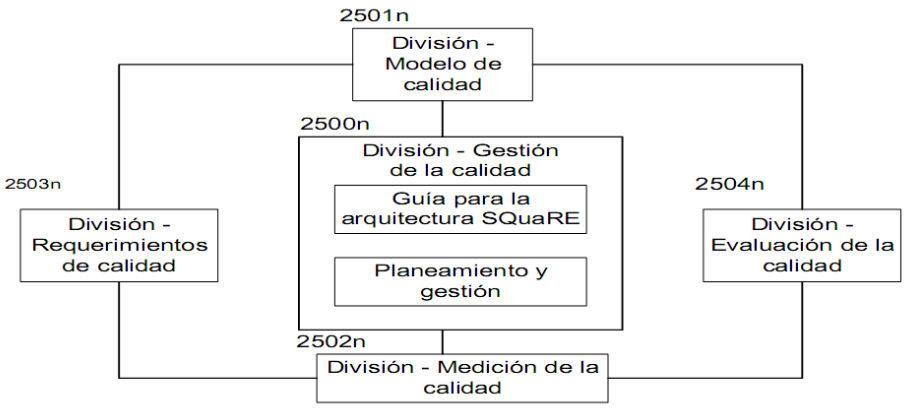
1.11.2.6. Parte 6: Documentación de los módulos de evaluación: Esta parte provee las guías para la documentación del módulo de evaluación. Estos módulos representan la especificación del modelo de calidad y las correspondientes métricas internas y externas que serán aplicadas a una evaluación en particular. Incluye métodos y técnicas de evaluación más las mediciones actuales resultantes de su aplicación. En esta parte también se considera la administración efectiva de complejidades inherentes a las cuestiones de medición.

Las actividades de medición coordinadas son una característica para una evaluación efectiva, además un plan necesita proveer un cronograma de evaluación que suministre al mismo tiempo información óptima cuando la evaluación sea conducida durante el desarrollo. Los módulos de la evaluación son componentes claves de la ISO/IEC 14598-6 y son usados para proveer un formato consistente y repetible de reporte. Dichos módulos proveen: Visibilidad de la información necesaria para cuadrar con requerimientos específicos de calidad, Documentación de las interfaces necesarias con herramientas de medición.

1.11.3. NORMA ISO/IEC 25000 (*SQUARE*): Desde el año 2000 se viene trabajando en el proyecto square que pretende establecer un modelo para la especificación y evaluación de un producto software, lo que ha llevado a reordenar la actual distribución de normas internacionales iso/iec 9126 e iso/iec 14598 y

considerando otras normas desarrolladas hasta la fecha. En la siguiente figura se puede apreciar la nueva arquitectura de la serie de normas 25000.

Figura 16. Arquitectura de la serie de normas ISO/IEC 25000 (tomado del módulo evaluación de software)



ISO 25000:2005 (SQuaRE -*Software Quality Requirements and Evaluation*) es una nueva serie de normas que se basa en ISO 9126 y en ISO 14598 (Evaluación del software). Uno de los principales objetivos de la serie SQuaRE es la coordinación y armonización del contenido de ISO 9126 y de ISO 15939:2002 (*Measurement Information Model*). ISO 15939 tiene un modelo de información que ayuda a determinar que se debe especificar durante la planificación, rendimiento y evaluación de la medición. Para su aplicación, cuenta con los siguientes pasos: Recopilar los datos, Preparación de los datos y Análisis de los datos.

La integración de ISO 9126 e ISO 15939 permiten plantear un proceso de 4 pasos: El primero, la identificación de los requerimientos relacionados a la calidad del producto, es decir, seleccionar la parte del modelo de calidad (ISO/IEC 9126-n) que resulta relevante para la evaluación de calidad; el segundo, la identificación del contexto de interpretación, es decir, selección de los valores de referencia y determinación de los target especificados en un contexto determinado; tercero, uso de las medidas derivadas de la etapa de preparación de los datos; y el cuarto, comparación y análisis de los resultados obtenidos respecto de un conjunto de valores de referencia.

SQuaRE incluye un estándar de requerimientos de calidad. Está compuesto por 14 documentos agrupados en cinco tópicos: Administración de la Calidad – 2500n,

Modelo de Calidad – 2501n, Medidas de Calidad – 2502n, Requerimientos de Calidad – 2503n y Evaluación de la Calidad – 2504n.

Administración de la Calidad: Abarca la Guía para SquaRE y Planificación y Administración.

Modelo de Calidad: Describe el modelo de calidad interno / externo y la calidad en uso, y presenta características y subcaracterísticas.

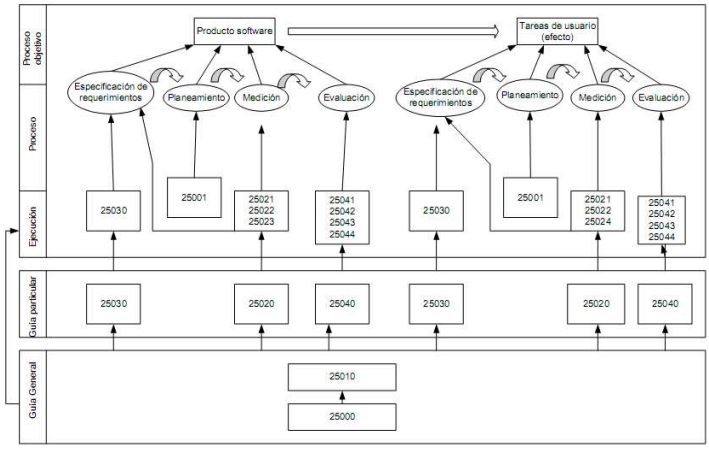
Medidas de Calidad: Medición de primitivas, Medidas para la calidad interna, Medidas para la calidad externa y Medidas para la calidad en uso.

Requerimientos de Calidad: Permite habilitar la calidad del software a ser especificado en términos de requerimientos de calidad durante todo el ciclo de vida de un proyecto de software o adquisición, mantenimiento y operación.

Evaluación de la Calidad: Evaluación de la Calidad, Proceso para desarrolladores, Proceso para compradores, Proceso para evaluadores y Documentación del módulo de evaluación.

En la figura siguiente se puede apreciar el modelo de referencia SquaRE, de la cual hay una reestructuración de los contenidos, se alinea a otros documentos existentes, y se amplían aspectos que en las normas anteriores sólo se señala a manera de información.

Figura 17. Modelo de referencia para la arquitectura Square (tomado del módulo evaluación de software)



1.12. RELACIÓN ENTRE MÉTRICAS INTERNAS Y EXTERNAS

Las métricas externas adecuadas y rangos aceptables son especificados para cuantificar los criterios de calidad que valida que el software satisface las necesidades del usuario. Los atributos de calidad interna del software, se definen y se especifica el plan para lograr finalmente la calidad requerida externa, la calidad en el uso y para construir en el producto durante el desarrollo. A demás, indicar unas métricas internas apropiadas y aceptables rangos especificados para cuantificar la calidad interna de atributos para que puedan ser utilizados para verificar que el software intermedio cumple con las especificaciones de calidad interno durante el desarrollo.

Se recomienda que las métricas internas que se utilizan tengan una relación tan fuerte como sea posible con los indicadores de destino externo, de modo que puedan ser utilizados para predecir los valores de las métricas externas.

Sin embargo, es generalmente difícil diseñar un modelo teórico riguroso que proporciona una fuerte relación entre indicadores internos y externos. [14]

1.13. ELECCIÓN DE MÉTRICAS Y CRITERIOS DE MEDICIÓN

La base sobre la cual se seleccionan los indicadores dependerá de los objetivos de negocio para el producto y las necesidades del evaluador. Las necesidades son especificadas por los criterios de medidas.

El modelo en esta parte del ISO / IEC 9126 soporta una variedad de requisitos de evaluación, por ejemplo:

• Un usuario o una unidad de un usuario de negocio puede evaluar la idoneidad de un producto de software utilizando métricas para la calidad en el uso.
• Un comprador podría evaluar un producto de software con los valores de criterio de las acciones exteriores de Funcionalidad, Confiabilidad, Usabilidad y Eficiencia, o de Calidad en uso.
• Un Administrador podría evaluar un producto de software utilizando métricas para el Mantenimiento.
• Una persona responsable de la implementación del software en distintos ambientes puede evaluar un producto de software utilizando métricas para la Portabilidad.

• Un desarrollador podría evaluar un producto de software con los valores de criterio orientados a las medidas internas de cualquiera de las características de calidad. [14]

1.14. MÉTRICAS UTILIZADAS PARA LA COMPARACIÓN

Al informar los resultados de la utilización de indicadores cuantitativos para hacer comparaciones entre los productos o con los valores de criterio, el informe deberá especificar si las métricas son objetivas, y los artículos empíricos con valor conocido y reproducible.

Comparaciones fiables, ya sea entre los productos o con los valores-criterio, sólo puede hacerse cuando se utilizan métricas rigurosas. Los procedimientos de medición debe medir la calidad de los productos de software, características (o subcaracterística) que dicen ser de una precisión suficiente para permitir que los criterios que se fijen y las comparaciones que se harán. Se tendrá en cuenta para posibles errores de medición causada por las herramientas de medición o error humano.

Métricas utilizadas para las comparaciones deben ser válidas y precisas, lo suficiente para permitir comparaciones fiables. Esto significa que las mediciones deben ser objetivas, empíricas utilizando una escala válida, y reproducibles.

• Para ser objetiva, habrá un procedimiento escrito y aprobado para la asignación del número o categoría para el atributo del producto.
• Para ser empírica, los datos se obtienen de la observación o un psicométricamente válida-cuestionario.
• Para utilizar una escala válida, los datos se basan en elementos de igual valor o de un valor conocido. Si una lista de verificación se utiliza para proporcionar los datos, los elementos de ser necesario se ponderarán.
• Para ser reproducibles, los procedimientos para la medida dará lugar a las mismas medidas (dentro de tolerancias apropiadas) se obtiene por diferentes personas haciendo la misma medida de productos de software en diferentes ocasiones.

Las métricas internas también deben tener una validez predictiva, es decir, deben correlacionarse con algunas acciones exteriores deseadas.

Por ejemplo, una medida interna de un atributo particular del software debe correlacionarse con algún aspecto mesurable de la calidad cuando se utiliza el software.

Es importante que mediciones para asignar valores coincidan con las expectativas normales, por ejemplo si la medición sugiere que el producto es de alta calidad, esto debería ser coherente con el producto en particular y satisfacer las necesidades del usuario. [14]

1.15. ITEMS QUE PUEDEN SER EVALUADOS

Los artículos pueden ser evaluados por la medición directa, o indirectamente mediante la medición de sus consecuencias. Por ejemplo, un proceso puede ser evaluado indirectamente mediante la medición y la evaluación es producto, y un producto puede ser evaluado indirectamente mediante la medición del rendimiento en la tarea de un usuario.
El software no se ejecuta solo, pero siempre como parte de un sistema mayor que suele estar compuesto de otros programas productos con los cuales tiene interfaces de hardware, operadores humanos, y flujos de trabajo. El producto de software completo puede ser evaluado por los niveles de los indicadores externos elegidos. Estas métricas describen su interacción con su entorno, y se evalúan mediante la observación de que el software está en funcionamiento.

La calidad en uso se puede medir por el grado en que un producto utilizado por usuarios específicos se ajuste a sus necesidades para alcanzar objetivos específicos con efectividad, productividad, seguridad y satisfacción. Esto normalmente se complementa con medidas de otras características de software específicas de calidad del producto, también es posible antes en el proceso de desarrollo.

En las primeras etapas de desarrollo, los recursos y el proceso sólo se pueden medir, cuando productos intermedios (especificaciones, código fuente, etc.) estén disponibles, éstos pueden ser evaluados por los niveles de las métricas internas elegidas. Estas métricas se pueden utilizar para predecir los valores de las métricas externas. También se puede medir por derecho propio, como requisitos previos esenciales para la calidad externa.

Otra distinción puede hacerse entre la evaluación de un producto de software y la evaluación del sistema en el que se ejecuta. [14]

1.16. UTILIZANDO UN MODELO DE CALIDAD

La calidad del producto de software debe ser evaluada mediante un modelo de calidad definido. El modelo de calidad debe indicar el tiempo para establecer objetivos de calidad para los productos de software y productos intermedios. La calidad de productos de software debe ser jerárquicamente en un modelo de calidad compuesto de las características y subcaracterísticas que pueden usarse como un cuestionario relacionadas con la calidad.

No es posible en la práctica medir todas las subcaracterísticas internas y externas de todas las partes de un producto de software de gran tamaño. Del mismo modo, no siempre es práctico medir la calidad en uso de todos los posibles escenarios. Los recursos para la evaluación deben ser repartidos entre los diferentes tipos de medida dependiendo de los objetivos de negocio, la naturaleza del producto y el diseño de procesos. [14]

1.17. EVALUACIÓN DE LA ARQUITECTURA DE SOFTWARE

La evaluación de software es como un estudio de factibilidad que pretende detectar posibles riesgos, y buscar recomendaciones para contenerlos, teniendo en cuenta que la evaluación se realiza antes de la implementación de la solución. El objetivo de evaluar una arquitectura es saber si puede habilitar los requerimientos, atributos de calidad y restricciones para asegurar que el sistema construido cumple con las necesidades de las personas que están relacionadas con el sistema.

La evaluación de una arquitectura de software es una tarea donde se pretende medir propiedades del sistema en base a especificaciones abstractas, como por ejemplo los diseños arquitectónicos. Las mediciones que se realizan sobre una arquitectura de software pueden tener distintos objetivos de tipo cualitativo, cuantitativo, o máximos y mínimos teóricos.

La medición cualitativa se aplica a la comparación entre arquitecturas candidatas y tiene relación con la intención de saber la opción que se adapta mejor a cierto atributo de calidad. La medición cuantitativa busca la obtención de valores que permitan tomar decisiones en cuanto a los atributos de calidad de una arquitectura de software, el esquema general es la comparación con márgenes establecidos, como lo son los requerimientos de desempeño, para establecer el grado de

cumplimiento de una arquitectura candidata, o tomar decisiones sobre ella. La medición de máximos y mínimos teórico contempla los valores teóricos para efectos de la comparación de la medición con los atributos de calidad especificados, el conocimiento de los valores máximos o mínimos permite el establecimiento claro del grado de cumplimiento de los atributos de calidad.

El propósito de realizar evaluaciones a la arquitectura, es para identificar y analizar riesgos potenciales en su estructura y sus propiedades, que afecten al software resultante, verificar que los requerimientos no funcionales estén presentes en la arquitectura, así como determinar el grado de satisfacción de los atributos de calidad. Un atributo de calidad es una característica de calidad que afecta a un elemento, donde el término “característica” se refiere a aspectos no funcionales y el termino “elemento” a componente.

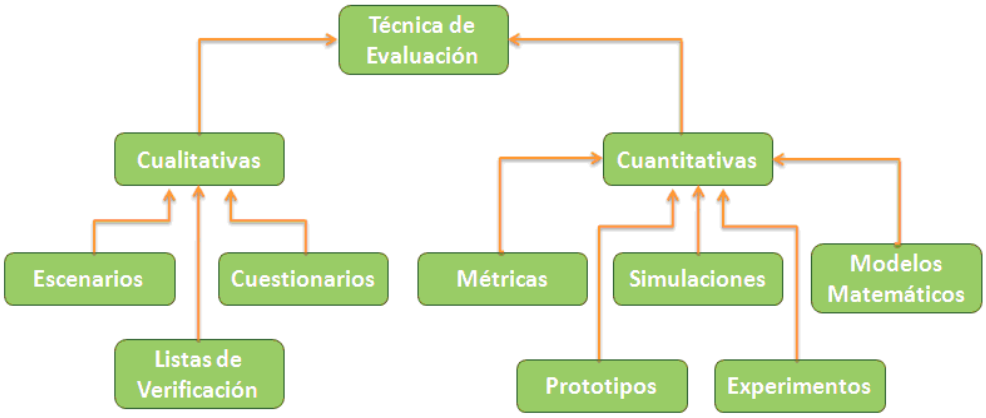
La evaluación de la arquitectura puede realizarse en cualquier momento, pero se propone dos etapas distintas para realizarlas: La evaluación previa que puede realizarse sin necesidad de que la arquitectura esté completamente especificada para efectuar la evaluación, abarca las fases iníciales de diseño y desarrollo, y la evaluación posterior que se hace cuando la arquitectura ya está establecida y la implementación se ha completado.

Generalmente las evaluaciones a la arquitectura se hacen por miembros del equipo de desarrollo, el arquitecto, el diseñador, entre otros. Otro actor interesado por los resultados de una evaluación es el cliente, ya que dependiendo de los resultados, puede tomar decisiones de que se continúe o no con el proyecto.

Las tánicas de evaluación de arquitecturas se clasifican en técnicas cualitativas y técnicas cuantitativas. Las técnicas cualitativas basadas en preguntas sobre la arquitectura (cuestionarios: abiertas y previas, *checklist*: específico del dominio de la aplicación, y escenarios: específicas del sistema, para arquitectura avanzada), y las cuantitativas que utiliza métricas como acoplamiento, cohesividad en los módulos, profundidad en herencias, modificabilidad.

Generalmente las técnicas de evaluación cualitativas son utilizadas cuando la arquitectura está en construcción, mientras que las técnicas de evaluación cuantitativas se usan cuando la arquitectura ya ha sido implantada. [7]

Figura 18. Clasificación de las técnicas de evaluación (tomado del módulo evaluación de software)



Las evaluaciones pueden ser planeadas o no planeadas: Las planeadas que son aquellas que han sido planificadas por el ciclo de vida de desarrollo, siendo parte de las actividades del proyecto, y las no planeadas que se presentan cuando la arquitectura contiene varios defectos que han sido detectados en las etapas tardías del desarrollo.

La arquitectura puede ser evaluada teniendo en cuenta la clasificación de los atributos de calidad, donde se hace la clasificación en dos categorías:

Observables vía ejecución que son aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución. La descripción de algunos de estos atributos se presenta en la siguiente tabla:

Tabla 6. Descripción de atributos de calidad observables vía ejecución

Atributo de calidad	Descripción
Disponibilidad	Es la medida de disponibilidad del sistema para el uso.
Confidencialidad	Es la ausencia de acceso no autorizado a la información.
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido.
Desempeño	Es el grado en el cual un sistema o componente cumple con las funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria
Confiabilidad	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo
Seguridad externa	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información

Seguridad interna	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos

No observables vía ejecución: aquellos atributos que se establecen durante el desarrollo del sistema. La descripción de algunos de estos atributos se presenta en la siguiente tabla.

Tabla 7. Descripción de atributos de calidad no observables vía ejecución

Atributo de calidad	Descripción
Configurabilidad	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema
Integrabilidad	Es la medida en que trabajan correctamente componentes del sistema al ser integrados, dado que fueron desarrollados separadamente.
Integridad	Es la ausencia de alteraciones inapropiadas de la información
Interoperabilidad	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema
Mantenibilidad	Es la capacidad de someter a un sistema a reparaciones y evolución
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos
Reusabilidad	Es la capacidad de diseñar un sistema, de forma tal, que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones
Escalabilidad	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental
Capacidad de Prueba	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas

1.18. TÉCNICAS DE EVALUACIÓN DE ARQUITECTURAS DE SOFTWARE

Las técnicas utilizadas para la evaluación de atributos de calidad requieren grandes esfuerzos del ingeniero de software para crear especificaciones y predicciones. Estas técnicas requieren información del sistema a desarrollar que

no está disponible durante el diseño arquitectónico, sino al principio del diseño detallado del sistema.

Las técnicas existentes en la actualidad para evaluar arquitecturas, permiten hacer una evaluación cuantitativa sobre los atributos de calidad a nivel arquitectónico, pero se tienen pocos medios para predecir los máximos o mínimos teóricos para las arquitecturas de software. Debido al costo de realizar este tipo de evaluación, en muchos casos los arquitectos de software evalúan cualitativamente, para decidir entre las alternativas de diseño. Se han propuesto diferentes técnicas de evaluación de arquitecturas de software, algunas son: evaluación basada en escenarios, evaluación basada en simulación, evaluación basada en modelos matemáticos y evaluación basada en experiencia. [7]

1.18.1. Evaluación basada en escenarios: Un escenario es una breve descripción de la interacción de alguno de los involucrados en el desarrollo del sistema. Por ejemplo, un usuario hará la descripción en términos de la ejecución de una tarea; un encargado de mantenimiento hará referencia a cambios que deban realizarse sobre el sistema; un desarrollador se enfocará en el uso de la arquitectura para efectos de su construcción o predicción de su desempeño.

Un escenario consta de tres partes: El estímulo, que es la parte del escenario que explica o escribe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema, incluye la ejecución de tareas, cambios en el sistema, ejecución de pruebas, configuración, etc; el contexto, que describe qué sucede en el sistema al momento del estímulo; y la respuesta que describe a través de la arquitectura, cómo debería responder el sistema ante el estímulo. Este último elemento es el que permite establecer cuál es el atributo de calidad asociado.

Los escenarios permiten concretar y entender atributos de calidad. Kazman y Carriere coinciden en la importancia del uso de los escenarios, no sólo para efectos de la evaluación de arquitecturas de software. Actualmente las técnicas basadas en escenarios cuentan con dos instrumentos de evaluación relevantes, a saber: el *Utility Tree* propuesto por Kazman, y los *Profiles* propuestos por Bosch.

1.18.1.1. Utility tree: Un *Utility Tree* es un esquema en forma de árbol que presenta los atributos de calidad de un sistema de software, refinados hasta el establecimiento de escenarios que especifican con suficiente detalle el nivel de prioridad de cada uno. La intención del uso del *Utility Tree* es la identificación de

los atributos de calidad más importantes para un proyecto particular. No existe un conjunto preestablecido de atributos, sino que son definidos por los involucrados en el desarrollo del sistema al momento de la construcción del árbol.

El *Utility Tree* contiene como nodo raíz la utilidad general del sistema. Los atributos de calidad asociados al mismo conforman el segundo nivel del árbol, los cuales se refinan hasta la obtención de un escenario lo suficientemente concreto para ser analizado y otorgarle prioridad a cada atributo considerado. Cada atributo de calidad perteneciente al árbol contiene una serie de escenarios relacionados, y una escala de importancia y dificultad asociada, que será útil para efectos de la evaluación de la arquitectura.

1.18.1.2. Perfiles (*Profiles*): Un perfil o *profile* es un conjunto de escenarios, generalmente con alguna importancia relativa asociada a cada uno de ellos. El uso de perfiles permite hacer especificaciones más precisas del requerimiento para un atributo de calidad. Los perfiles tienen asociados dos formas de especificación:
Los perfiles completos que definen los escenarios importantes como parte del perfil. Esto permite al ingeniero de software realizar un análisis de la arquitectura para el atributo de calidad estudiado de una manera completa, puesto que incluye todos los posibles casos. Su uso se reduce a sistemas relativamente pequeños y sólo es posible predecir conjuntos de escenarios completos para algunos atributos de calidad.

Los perfiles seleccionados que se asemejan a la selección de muestras sobre una población en los experimentos estadísticos. Se toma un conjunto de escenarios de forma aleatoria, de acuerdo a algunos requerimientos. La aleatoriedad no es totalmente cierta por limitaciones prácticas, por lo que se fuerza la realización de una selección estructurada de elementos para el conjunto de muestra. Si bien es informal, permite hacer proposiciones científicamente válidas.
Para la definición de un perfil, es necesario seguir tres pasos: definición de las categorías del escenario, selección y definición de los escenarios para la categoría y asignación del “peso” a los escenarios. La definición de categorías de escenarios divide la población de escenarios en poblaciones más pequeñas, que cubren aspectos particulares del sistema. La selección y definición de escenarios para cada categoría selecciona un conjunto de escenarios representativo para la subpoblación. Luego, en la asignación del peso a los escenarios, dependiendo del perfil, el peso de un escenario tiene diferentes significados. Se definen escalas

que de alguna forma sean cuantificables y puedan ser convertidas a pesos relativos.

Cada atributo de calidad tiene un perfil asociado. Algunos perfiles pueden ser usados para evaluar más de un atributo de calidad. Han sido seleccionados cinco atributos de calidad que son considerados de mayor relevancia para una perspectiva general de ingeniería de software, ellos son: desempeño *(performance)*, mantenibilidad *(maintainability)*, confiabilidad *(reliability)*, seguridad externa *(safety)* y seguridad interna *(security)*.

La efectividad de la técnica es altamente dependiente de la representatividad de los escenarios. Existe la evaluación de funcionalidad basada en escenarios, y es utilizada en el diseño orientado a objetos para especificar comportamiento del sistema. La diferencia entre este tipo de evaluación y la evaluación arquitectónica basada en escenarios radica en que la última utiliza los escenarios para evaluar atributos de calidad, en lugar de verificar o describir funcionalidad, además de que se usan otros escenarios para definir otros atributos de calidad. La siguiente tabla muestra para cada atributo de calidad, el perfil asociado, la forma en que se definen las categorías, el significado de los "pesos" y posibles métricas de evaluación. [7]

Tabla 8.Perfiles, categorías, pesos, y métricas asociados a atributos de calidad

Atributo	**Perfil**	**Categorías**	**Pesos**	**Métricas**
Mantenibi-lidad	Perfil de Manteni-miento (*Maintainanc e profile*)	Se organizan alrededor de las interfaces del sistema (sistema operativo, interfaces con el usuario, interfaces con otros sistemas). Los escenarios de cambio describen modificaciones en los requerimientos	Indican la probabilidad de ocurrencia del cambio de escenario en un período de tiempo	- Impacto en término de líneas de código que tienen que cambiarse - Se requiere un estimado de líneas de código de los componentes arquitectónicos.

Desempe-ño	Perfil de uso (*Usage profile*)	Descompone los escenarios de uso basado en los tipos de usuario y/o interfaces del sistema	Representan la frecuencia relativa del escenario	- Funcionalidad de componentes - Comportamiento del sistema en respuesta a los escenarios de uso en el perfil - Promedio y peor caso de latencia por sincronización y sobrecarga en el sistema
Confiabili-dad	Perfil de uso (*Usage profile*)	Confiabilidad de los componentes, genera la confiabilidad de los escenarios de uso	Indica la robustez del sistema	- Datos estimados de confiabilidad del componente - Datos históricos de confiabilidad del componente
Seguridad Interna	Perfil de seguridad (*Security profile*) Perfil de uso (*usage profile*)	Basada en todas las interfaces del sistema	Indica la probabilidad de fallas	Depende del aspecto de seguridad a ser evaluado. Por ejemplo, la disponibilidad puede evaluarse en términos del número de veces que se ejecutan operaciones de seguridad.
Seguridad Externa	Perfil de peligro (*Hazard profile*)	Se organizan de acuerdo a documentos de certificación (puntos de interacción del sistema con el mundo real, o componentes	Indican la probabilidad de falla u ocurrencia e consecuen-cias desastrosas.	*Bosch (2000) no establece ejemplos*

		críticos del sistema)		

La evaluación basada en escenarios puede ser empleada para comparar dos arquitecturas y para la evaluación absoluta de una arquitectura. La diferencia está en que la evaluación absoluta requiere mayor cantidad de datos estimados y cuantitativos necesarios para la evaluación.

La técnica se puede descomponer en dos etapas: El análisis de impacto, que toma como entrada el perfil y la arquitectura de software, para cada escenario del perfil, se evalúa el impacto de la arquitectura y se obtienen los resultados. La etapa de predicción de resultados, donde los resultados serán usados para pronosticar el valor del atributo de calidad estudiado de acuerdo a las métricas existentes. [7]

1.18.1.3. Evaluación basada en simulación: La evaluación basada en simulación utiliza una implementación de alto nivel de la arquitectura de software. El enfoque básico consiste en la implementación de componentes de la arquitectura y la implementación del contexto del sistema donde se supone va a ejecutarse. La finalidad es evaluar el comportamiento de la arquitectura bajo diversas circunstancias. Una vez disponibles estas implementaciones, pueden usarse los perfiles respectivos para evaluar los atributos de calidad. El proceso de evaluación basada en simulación sigue los pasos mostrados en la siguiente tabla:

Tabla 9. Pasos para la evaluación basada en simulación

Pasos	Definición
Definición e implementación del contexto	Consiste en identificar las interfaces de la arquitectura de software con su contexto, y decidir cómo será simulado el comportamiento del contexto en tales interfaces.
Implementación de los componentes arquitectónicos	La descripción del diseño arquitectónico debe definir las interfaces y las conexiones de los componentes, por lo que estas partes pueden ser tomadas directamente de la descripción de diseño.
Implementación de los componentes arquitectónicos	La descripción del diseño arquitectónico debe definir las interfaces y las conexiones de los

	componentes, por lo que estas partes pueden ser tomadas directamente de la descripción de diseño.
Implementación del perfil	Dependiendo del atributo de calidad que el arquitecto de software intenta evaluar usando simulación, el perfil asociado necesitará ser implementado en el sistema. El arquitecto de software debe ser capaz de activar escenarios individuales, así como también ejecutar un perfil completo usando selección aleatoria, basado en los pesos normalizados de los mismos.
Simulación del sistema e inicio del perfil	El arquitecto de software ejecutará la simulación y activará escenarios de forma manual o automática, y obtendrá resultados de acuerdo al atributo de calidad que está siendo evaluado.
Predicción de atributos de calidad	Dependiendo del tipo de simulación y del atributo de calidad evaluada, se puede disponer de cantidades excesivas de datos, que requieren ser condensados. Esto permite hacer conclusiones acerca del comportamiento del sistema.

En el ámbito de las simulaciones, se encuentra la técnica de implementación de prototipos. Esta técnica implementa una parte de la arquitectura de software y la ejecuta en el contexto del sistema. Es utilizada para evaluar requerimientos de calidad operacional, como desempeño y confiabilidad. [7]

1.18.1.4. Evaluación basada en modelos matemáticos: La evaluación basada en modelos matemáticos se utiliza para evaluar atributos de calidad operacionales. Permite una evaluación estática de los modelos de diseño arquitectónico, y se presentan como alternativa a la simulación, dado que evalúan el mismo tipo de atributos. Ambos enfoques pueden ser combinados, utilizando los resultados de uno como entrada para el otro. El proceso de evaluación basada en modelos matemáticos sigue los pasos mostrados en la siguiente tabla:

Tabla 10. Pasos para la evaluación basada en modelos matemáticos

Pasos	Definición
Selección y adaptación del modelo matemático	Existen varios modelos matemáticos para medir sus atributos de calidad, los cuales tienden a ser muy elaborados y detallados, así como también requieren de cierto tipo de datos y análisis. Parte de estos datos requeridos no están disponibles a nivel de arquitectura, y la técnica requiere mucho esfuerzo para la evaluación arquitectónica, por lo cual se debe adaptar el modelo.
Representación de la arquitectura en términos del modelo	El modelo matemático seleccionado y adaptado no asume necesariamente que el sistema que intenta modelar consiste de componentes y conexiones. Por lo tanto, la arquitectura necesita ser representada en términos del modelo.
Estimación de los datos de entrada requeridos	El modelo matemático aún cuando ha sido adaptado, requiere datos de entrada que no están incluidos en la definición básica de la arquitectura. Es necesario estimar y deducir estos datos de la especificación de requerimientos y de la arquitectura diseñada.
Predicción de atributos de calidad	Una vez que la arquitectura es expresada en términos del modelo y se encuentran disponibles todos los datos de entrada requeridos, el arquitecto está en capacidad de calcular la predicción resultante del atributo de calidad evaluado.

Entre los instrumentos que se cuentan para las técnicas de evaluación de arquitecturas de software basada en modelos matemáticos, se encuentran las Cadenas de Markov y los denominados "*Reliability Block Diagrams"*. [7]

1.18.1.5 Evaluación basada en experiencia: En muchas ocasiones los arquitectos e ingenieros de software otorgan valiosas ideas que resultan de utilidad para evitar decisiones erradas de diseño. Aunque todas estas experiencias

se basan en factores subjetivos como la intuición y la experiencia. Sin embargo, la mayoría de ellas puede ser justificada por una línea lógica de razonamiento, y pueden ser la base de otros enfoques de evaluación.

Existen dos tipos de evaluación basada en experiencia: la evaluación informal, que es realizada por los arquitectos de software durante el proceso de diseño, y la realizada por equipos externos de evaluación de arquitecturas. La siguiente tabla muestra los instrumentos utilizados por las diferentes técnicas de evaluación. [7]

Tabla 11. Instrumentos asociados a las distintas técnicas de evaluación de arquitecturas de software

Técnica de Evaluación	**Instrumento de Evaluación**
Basada en Escenarios	- *Profiles* - *Utility Tree*
Basada en Simulación	- Lenguajes de Descripción Arquitectónica (ADL) - Modelos de colas
Basada en Modelos Matemáticos	- Cadenas de Markov - *Reliability Block Diagrams*
Basada en Experiencia	- Intuición y experiencia - Tradición - Proyectos similares

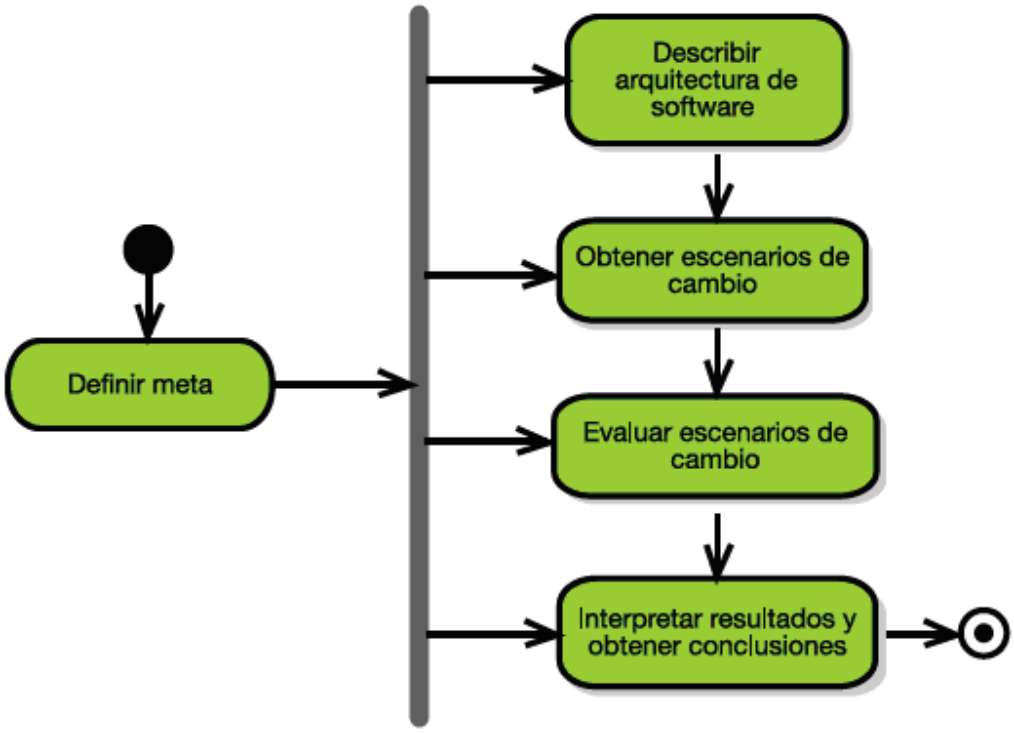
1.19. MÉTODOS DE EVALUACIÓN DE ARQUITECTURA DE UN ATRIBUTO ESPECÍFICO

Existen otros métodos de evaluación de arquitectura del software, que evalúan solamente uno de los atributos de calidad especificado por el evaluador, algunos de ellos son:

1.19.1. Architecture Level Modifiability Analysis ALMA: Este método es el resultado de los trabajo de investigación de Brengtsson y Lassing, el atributo de calidad que analiza este método es la facilidad de modificación, este atributo se refiere a la capacidad de un sistema para ser ajustado debido a cambios en los requerimientos, o en el entorno, así como la adición de nuevas funcionalidades.

ALMA es un método de evaluación orientado a metas, que se apoya en el uso de escenarios de cambio, los cuales escriben los eventos posibles que provocarían cambios al sistema, y como se llevarían a cabo estos. El mismo consta de cinco pasos como se muestran en la siguiente figura: [7]

Figura 19. Método de evaluación de arquitecturas ALMA (tomado del módulo evaluación de software)

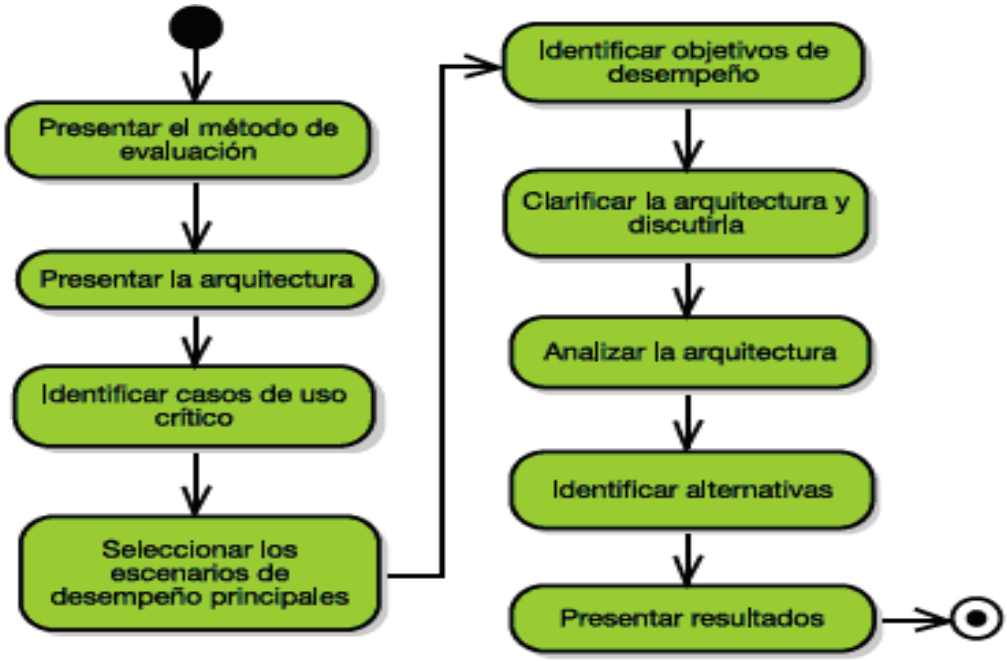


1.19.2. Performance Assessment of Software Architecture PASA: Es el resultado del trabajo de Williams y Smith, el mismo utiliza diversas técnicas de evaluación, tales como la aplicación de estilos arquitectónicos, anti-patrones, guías de diseño y modelos. El atributo de calidad que analiza este método es el desempeño, se interesa en saber qué tanto tiempo le toma al sistema de software responder cuando una o varios eventos ocurren, así como determinar el número de eventos procesados en un intervalo de tiempo dado.

Este método también se basa en escenarios y puede aplicarse de forma temprana o tardía. Los escenarios generados en este método sirven como punto de partida para la construcción de modelos de desempeño. Uno de los requisitos o

precondiciones es que la arquitectura debe estar previamente documentada y en caso de que no esté completa, se debe extraer el resto de la información a los miembros del equipo.

Figura 20. Método de evaluación de arquitecturas PASA (tomado del módulo evaluación de software)

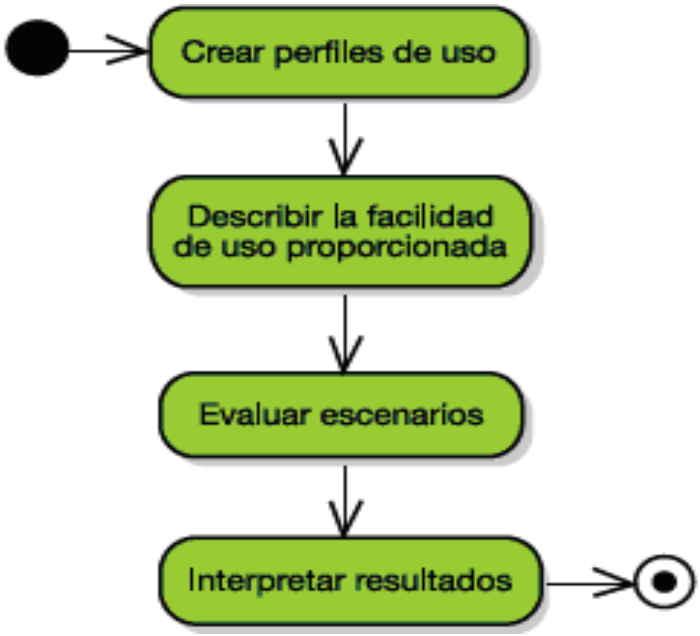


1.19.3. Scenario based Architecture Level Usability Analysis SALUTA: Es el primer método desarrollado para evaluar arquitecturas desde la perspectiva de la facilidad del uso del sistema, siendo el resultado de los estudios de Folmer y Gurp. Este método hace uso de marcos de referencia que expresan las relaciones que existen entre facilidad de uso y Arquitectura de Software. Dichos marcos de referencias incluyen un conjunto integrado de soluciones de diseño como patrones de diseños u propiedades que tienen un efecto positivo sobre la facilidad de uso en un sistema de software.

SALUTA analiza cuatro atributos que están directamente relacionados con la facilidad de uso de un sistema de software: facilidad de aprendizaje, eficiencia de uso, confiabilidad y satisfacción. El método se basa en escenarios, que en este caso, son escenarios de uso que agrupan uno a más perfiles de uso, valga la redundancia, donde cada uno representa la facilidad de uso requerida por el

sistema. Se recomienda utilizarlo una vez que se ha especificado la arquitectura, pero antes de implementarla. La siguiente figura muestra los cuatro pasos del método: [7]

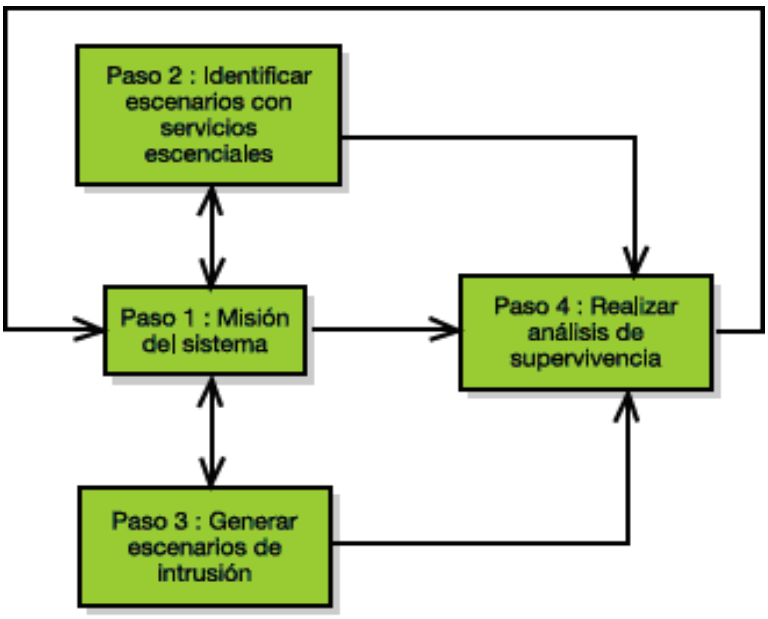
Figura 21. Método de evaluación de arquitecturas SALUTA (tomado del módulo evaluación de software)



1.19.4. Survivable Network Analysis SNA: Es un método desarrollado por el CERT (*Computer Emergency Response Team*) que forma parte del SEI (*Software Engineering Institute*). Este método ayuda a identificar la capacidad de supervivencia en un sistema, analizando su arquitectura. La supervivencia es la capacidad que tiene un sistema para completar su misión a tiempo, ante la presencia de ataques, fallas o accidentes. Para evaluar esta supervivencia SNA utiliza tres propiedades claves: resistencia, reconocimiento y recuperación.

Este procedimiento puede ser realizado después de la especificación de la arquitectura, durante la implementación de esta, o posteriormente. SNA se basa en la técnica de escenarios de uso, e identifica dos tipos de escenarios: escenarios normales, que se componen de una serie de pasos donde los usuarios invocan servicios y obtienen acceso a activos, tales como bases de datos; escenarios de intrusión, en los que se representan diferentes tipos de ataques al sistema. En la siguiente figura se muestra cada uno de los pasos del método: [7]

Figura 22. Método de evaluación de arquitecturas SNA (tomado del módulo evaluación de software)



La siguiente tabla muestra comparativamente de cada uno de los métodos de evaluación de arquitectura mencionados anteriormente:

Tabla 12. Comparación entre los métodos ALMA, PASA, SALUTA y SNA.

	ALMA	**PASA**	**SALUTA**	**SNA**
Meta	Predecir el costo de mantenimiento, evaluar riesgos, comparación entre arquitecturas	Analizar la arquitectura con respecto a los objetivos de desempeño de un sistema	Predecir la facilidad de uso de un sistema analizando la arquitectura	Identificar la capacidad de supervivencia de un sistema ante presencia de ataques, fallas o accidentes
Atributo de calidad	Facilidad de modificación	desempeño	Facilidad de uso	supervivencia
Técnica de evalua-ción	Escenarios de cambio	escenarios	Escenarios de uso	Escenarios de uso normales y escenarios de intrusión
entradas	Especificación de la arquitectura,	Especificación de la arquitectura	Especificación de la arquitectura,	Especificación de la arquitectura

	requerimientos no funcionales		requerimien-tos no funcionales relacionados con la facilidad de uso	
salidas	Dependiendo de la meta de evaluación se generan los resultados	Hallazgos encontrados, pasos específicos a seguir y recomendacio-nes	Grado de facilidad de uso que soporta la arquitectura evaluada	Modificaciones recomendadas de la arquitectura y mapa de supervivencia
Personas involucra-das	Arquitecto y equipo de desarrollo	Arquitecto, equipo de desarrollo y administradores del proyecto	Arquitecto, ingenieros de requerimien-tos o ingenieros responsables por la facilidad de uso	Arquitecto, diseñador principal, propietarios del sistema, usuarios
Duración	No especificado	7 días	No especificado	No especificado
Validación del método	Sistemas de control embebido, sistemas médicos, tele-comunicacione s, sistemas administrativos	Sistemas basados en la web, aplicaciones financieras, y sistemas en tiempo real	Algunos casos de estudio que incluyen principalmente sistemas web	Sistemas comerciales y de gobierno

1.20. LENGUAJE UNIFICADO DE MODELADO

El lenguaje unificado de modelado es una consolidación de muchas de las notaciones y conceptos orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

En 1996, el *Object Management Group* (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una solicitud con propósito de una meta modelo orientado a objetos de semántica y notación estándares.

UML prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, es ideal que los modeladores sólo tengan que aprender una única notación. [23]

1.20.1. Diagrama de casos de uso: Un caso de uso especifica el comportamiento del sistema o parte de él en relación con la intervención de un usuario, define un conjunto de acciones que se realizan para generar un resultado observable e importante para un actor.

Los límites del sistema se indican mediante un cuadro o un rectángulo, dentro de éste se coloca los casos de uso y fuera, los actores. Los casos de uso se representan mediante elipses y los actores mediante figuras humanas.

Un actor representa un rol desempeñado por algún usuario, dispositivo o sistema externo que interactúa con el sistema en estudio. Los casos de uso ayudan a identificar y modelar los requisitos funcionales del sistema, por este motivo son ampliamente utilizados en las primeras fases del proceso de desarrollo de software. Los casos de uso pueden organizarse especificando relaciones entre ellos como: especializaciones, inclusiones y extensiones.

La especialización o relación de generalización en los casos de uso tiene el mismo significado y funcionalidad que en las clases. Dado un caso de uso base, puede tenerse otro especializado (hijo) que hereda el comportamiento del primero. El caso de uso especializado puede añadir funciones o redefinir las que ha heredado. La representación de la generalización entre casos de uso se hace de igual manera que en las clases, mediante una línea continua terminada en punta de flecha vacía que se dirige desde el caso de uso especializado hasta el caso de uso general.

Las relaciones de inclusión se utilizan para evitar describir el mismo flujo de eventos repetidas veces, poniendo el comportamiento común en un caso de uso aparte. La inclusión consiste en delegar funciones.
Se identifica responsabilidades del sistema que figuren en más de un caso de uso, se elabora un caso de uso para dichas funciones y luego se incluye en otros casos de uso que requieran desarrollar dichas funciones. Para indicar una relación de inclusión entre casos de uso se utiliza una dependencia con el estereotipo *include*.

Una relación de extensión indica que el comportamiento de un caso de uso base puede extenderse con el comportamiento de otro caso de uso que extiende al primero. Este tipo de relaciones se representan como una dependencia estereotipada con la palabra *extend*.

1.20.2. Diagramas de secuencia: Es un diagrama para mostrar una interacción entre un conjunto de objetos indicando el orden en que se envían y reciben los mensajes.

Para elaborar el diagrama de secuencia se colocan en la parte superior, siguiendo el eje X, los objetos que participan en la interacción, para ello se debe tener en cuenta colocar a la izquierda el objeto que inicia la interacción y los objetos subordinados hacia la derecha. Debajo de cada objeto y siguiendo el eje Y, se traza la línea de vida y el foco de control.

La línea de vida. Es una línea discontinua vertical, trazada debajo de un objeto e indica la existencia de éste en el tiempo. Algunos objetos existen durante toda la interacción, en cuyo caso el objeto se colocará en la parte superior del diagrama y la línea de vida se trazará de arriba hacia abajo hasta el fin del diagrama. Algunos objetos serán creados y destruidos durante la interacción, en este caso el objeto se colocará a la altura del mensaje que lo crea y su línea de vida se extenderá a partir de esa posición.

El foco de control. Es un rectángulo delgado que se traza de forma vertical sobre la línea de vida para indicar el tiempo que el objeto está ejecutando una acción, ya sea que ejecute la acción con sus propios métodos o con métodos subordinados, es decir, invocando métodos en otros objetos. La parte superior del foco de control indica el comienzo de la acción y debe estar a la altura del primer mensaje, mientras que la parte inferior corresponde al fin de la acción y puede marcarse con un mensaje de retorno. [15]

1.21. METODOLOGIA RUP

RUP (*Rational Unified Process*) se basa en un conjunto de bloques de construcción, o los elementos de contenido, describiendo lo que se va a producir, las habilidades necesarias y la explicación paso a paso que describe cómo poder alcanzar los objetivos específicos de desarrollo.

El RUP ha determinado un ciclo de vida del proyecto que consta de cuatro fases. Estas fases permiten que el proceso que se presentará en un alto nivel de una manera similar a como un proyecto en cascada presentado, aunque, en esencia, la clave del proceso radica en las iteraciones de desarrollo que se encuentran dentro de todas las fases. Además, cada fase tiene un objetivo fundamental y un hito en el extremo que denota el objetivo cumplido.

Fase Inicial

El objetivo principal es el alcance del sistema de forma adecuada como base para la validación inicial de costos y presupuestos. En esta fase el caso de negocios que incluye el contexto empresarial, factores de éxito (los ingresos previstos, el reconocimiento del mercado, etc.), y las previsiones financieras se establece. Para complementar el caso de negocios se generan, un modelo de casos de uso básico, plan del proyecto, la evaluación inicial de riesgos y descripción del proyecto (los requisitos del proyecto básico, las limitaciones y características principales). Después de estos requisitos se han completado, el proyecto se comprueba con los siguientes criterios:

- Las partes interesadas acuerdo sobre la definición del alcance y el costo / estimaciones de lo previsto.
- Descripción de los requisitos como lo demuestra la fidelidad de los casos de uso principales.
- La credibilidad de la relación coste / estimaciones calendario, las prioridades, los riesgos, y el proceso de desarrollo.
- Profundidad y amplitud de cualquier prototipo de la arquitectura que se desarrolló.
- El establecimiento de una línea de base con la que comparar los gastos reales frente a los gastos previstos.

Si el proyecto no pasa este hito, llamado el ciclo de vida del Objetivo Milestone, puede ser cancelado o se repite después de ser rediseñado para responder mejor a los criterios.

Fase de elaboración

El objetivo principal es mitigar los elementos de riesgo identificados por el análisis hasta el final de esta fase. La fase de elaboración es cuando el proyecto empieza

a tomar forma. En esta fase, se hace el análisis de dominio del problema y la arquitectura del proyecto toma su forma básica.
Esta fase debe pasar el hito de ciclo de vida de Arquitectura al cumplir con los siguientes resultados:

- Un modelo de casos de uso, en el que los casos de uso y los actores han sido identificados y la mayoría de las descripciones de casos de uso desarrollados. El modelo de caso de uso debe ser del 80% de avance.
- Una descripción de la arquitectura de software, en un proceso de desarrollo de software del sistema.
- Una arquitectura ejecutable, que se da cuenta de casos de uso arquitectónicamente significativos.
- Caso de negocio y la lista de riesgos, que se revisan.
- Un plan de desarrollo para el proyecto en general.
- Prototipos que demuestren una técnica para mitigar cada riesgo identificado.

Si el proyecto no puede pasar este hito, aún hay tiempo para que sea cancelado o rediseñados. Sin embargo, después de salir de esta fase, el proyecto de transición es una operación de alto riesgo donde los cambios son mucho más difíciles y perjudiciales que cuando se implementan los casos de uso.

El análisis de dominio es clave para la elaboración de la arquitectura del sistema.

Fase de Construcción

El objetivo principal es construir el sistema de software. En esta fase, la atención se centra en el desarrollo de componentes y otras características del sistema. Esta es la fase en que la mayor parte de la codificación se lleva a cabo. En proyectos más grandes, de varias iteraciones de construcción pueden ser desarrollados en un esfuerzo por dividir a los casos de uso en segmentos manejables que producen prototipos demostrables.
Esta fase produce la primera versión externa del software. Su conclusión está marcada por el hito inicial de la Capacidad Operacional.

Fase de Transición

El objetivo principal es "tránsito" del sistema de desarrollo en producción, puesta a disposición y entendido por el usuario final. Las actividades de esta fase incluyen

la formación de los usuarios finales y mantenedores, y las pruebas beta del sistema para validar las expectativas de los usuarios finales. El producto también se comprueba con el nivel de calidad fijados en la fase de iniciación.
Si todos los objetivos se cumplen, el hito de lanzamiento del producto se alcanza y termina el ciclo de desarrollo. [16]

1.22. HERRAMIENTAS PARA LA CONSTRUCCION DEL APLICATIVO INFORMATICO QUALITYSOFT

1.22.1. Lenguaje de programación C sharp (c#): C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, el cual es similar al de Java, aunque incluye mejoras derivadas de otros lenguajes (entre ellos Delphi).

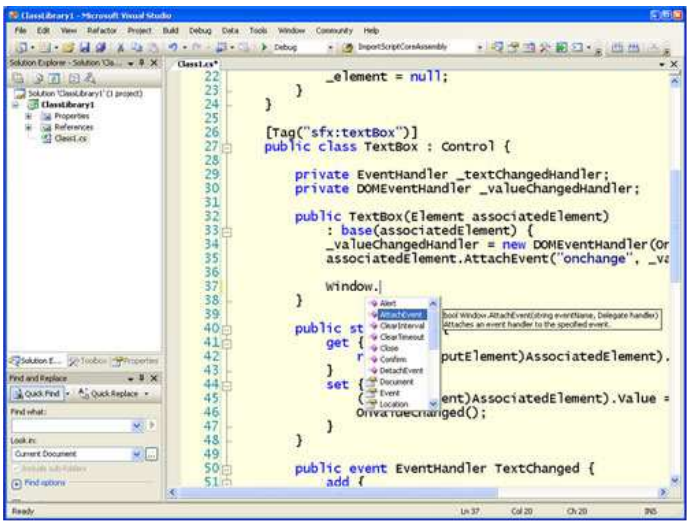
La creación del nombre del lenguaje C#, proviene de dibujar dos signos positivos encima de los dos signos positivos de "C++", queriendo dar una imagen de salto evolutivo del mismo modo que ocurrió con el paso de C a C++.

C#, como parte de la plataforma .NET, está normalizado por ECMA desde diciembre de 2001 (ECMA-334 "Especificación del lenguaje C#"). El 7 de noviembre de 2005 salió la versión 2.0 del lenguaje que incluía mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales y tipos anulables.
Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones (API); mientras que C# es un lenguaje de programación independiente, diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco de DotGNU - Mono que genera programas para distintas plataformas como Win32, UNIX y Linux. [17]

C# combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, Visual Basic o Delphi. De hecho, su creador AndersHeljsberg fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal,

Delphi o Visual J++. La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, y que además la migración a este lenguaje por los programadores de C/C++/Java sea lo más inmediata posible. [18]

Figura 23. C Sharp



Características de C Sharp. "Sencillez: C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:
El código escrito en C# es auto contenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL.

El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para los que se compile (no como en C++), esto facilita la portabilidad del código.

No se incluyen elementos poco útiles de lenguajes como C++ tal es el caso de macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::)

Modernidad: C# incorpora en el propio lenguaje elementos que a lo largo de los años se ha demostrado que son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, tales como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de

128 bits (muy útil en el mundo financiero), la inclusión de una instrucción *foreach* que permita recorrer arreglos con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico *string* para representar cadenas o la distinción de un tipo *bool* específico para representar valores lógicos.

Orientación a objetos: Como todo lenguaje de programación actual de propósito general, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del *Common Type System* que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++, es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.
C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo. En lo referente a la encapsulación, es importante señalar que aparte de los típicos modificadores *public, private y protected*, C# añade un cuarto modificador llamado internal, que puede combinarse con *protected* e indica que al elemento a cuya definición precede, sólo puede accederse desde su mismo ensamblado.

Orientación a componentes: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros)
Gestión automática de memoria: Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del *Common Language Runtime*. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active (ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente), C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción *using*.

Seguridad de tipos: C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evitar que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacene una referencia del segundo (*downcasting*) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.

No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control de la fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.

A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.

Instrucciones seguras: Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=).
Sistema de tipos unificado: A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada *System.Object*, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán “objetos”)

Extensibilidad de tipos básicos: C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a la pila a través del modificador de parámetro ref.

Extensibilidad de operadores: Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores (incluidos los de conversión, tanto para conversiones implícitas como explícitas) cuando se apliquen a diferentes tipos de objetos.

Versionable: C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién creados.

Eficiente: En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad muy alta de procesamiento.

Compatible: Para facilitar la migración de programas, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código fuente en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados *PlatformInvocationServices* (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. [19]

1.22.2. Microsoft visual studio 2010: Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio 2010 es la versión más reciente de esta herramienta, acompañada por .NET Framework 4.0. La fecha prevista para el lanzamiento de la versión final fue el 12 de abril de 2010.

Hasta ahora, uno de los mayores logros de la versión 2010 de Visual Studio ha sido el de incluir las herramientas para desarrollo de aplicaciones para Windows 7, tales como herramientas para el desarrollo de las características de Windows 7 (*System.Windows.Shell)* y la *Ribbon Preview* para *Windows Presentation Foundation*.

Entre sus más destacables características, se encuentran la capacidad para utilizar múltiples monitores, así como la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo. Además de esto, aparece una edición que compila las características de todas las ediciones comunes de Visual Studio: Professional, *Team Studio*, Test, conocida como *Visual Studio Ultimate*

A las mejoras de desempeño, escalabilidad y seguridad, se agregan entre otras, las siguientes novedades:

La mejora en las capacidades de Pruebas Unitarias, permiten ejecutarlas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos. Se incluye además un nuevo soporte para diagnosticar y optimizar el sistema a través de las herramientas de pruebas de Visual Studio. Con ellas se podrán ejecutar perfiles durante las pruebas para que ejecuten cargas, prueben procedimientos contra un sistema y registren su comportamiento; y utilizar herramientas integradas para depurar y optimizar. [20]

1.22.3. Mysql 5.1: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario.Existen varias APIs que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C#, Lisp, Perl, PHP, Python, Ruby, Gambas, REALbasic (Mac y Linux); cada uno de estos utiliza una API específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. Además se puede acceder desde el sistema SAP, lenguaje ABAP.

En MySQL 5.1, se puede almacenar cada tabla InnoDB y sus índices en su propio fichero. Esta característica se llama "*multiple tablespaces*" (espacios de tablas múltiples) porque, en efecto, cada tabla tiene su propio espacio de tablas.

El uso de múltiples espacios de tablas puede ser beneficioso para usuarios que desean mover tablas específicas a discos físicos separados o quienes deseen restaurar respaldos de tablas sin interrumpir el uso de las demás tablas InnoDB.

1.22.3.1 Características: Proporciona sistemas de almacenamiento transaccional y no transaccional. Uso completo de *multi-threaded* mediante *threads* del *kernel*. Pueden usarse fácilmente múltiple CPUs si están disponibles.

Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.

Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

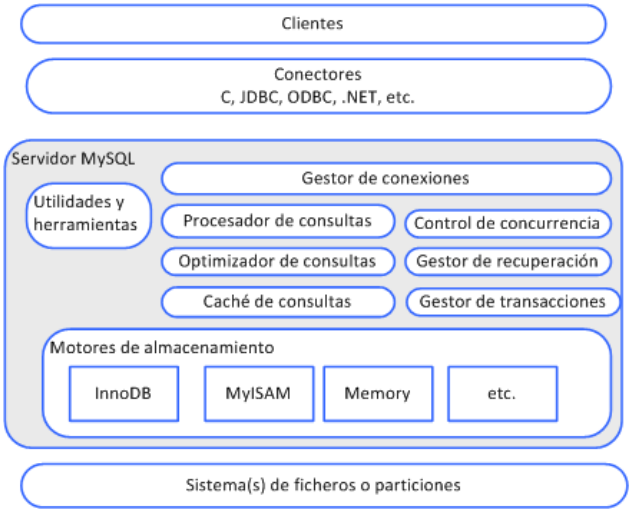
En MySQL 5.0, los servidores Windows soportan conexiones con memoria compartida si se inicializan con la opción *--shared-memory*. Los clientes pueden conectar a través de memoria compartida usando la opción *--protocol=memory*.

La interfaz para el conector ODBC (MyODBC) proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (*Open Database Connectivity*). Por ejemplo, puede usar MS Access para conectar al servidor MySQL. Los clientes pueden ejecutarse en Windows o Unix. El código fuente de MyODBC está disponible. Todas las funciones para ODBC 2.5 están soportadas, así como muchas otras.

MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas. Estos comandos están disponibles a través de la línea de comandos y el cliente mysqlcheck. MySQL también incluye myisamchk, una utilidad de línea de comandos muy rápida para efectuar estas operaciones en tablas MyISAM(Transaccionales).[21]

1.22.3.2. Arquitectura mysql: La arquitectura de MYSQL tiene como característica más notable el separar el motor de almacenamiento (que se encarga de los detalles de entrada-salida y representación de la información en memoria secundaria) del resto de los componentes de la arquitectura. Es decir, el diseño del gestor está preparado para que se pueda cambiar los atributos de almacenamiento. Esto permite incluso crear nuevos motores de almacenamiento especializados para ciertas tareas o tipos de aplicaciones.

Figura 24. Arquitectura MySql



Motores de almacenamiento

El elemento más notable de la arquitectura de MySQL es la denominada arquitectura de motores de almacenamiento reemplazables (*pluggable storage engine architecture*). La idea de esa arquitectura es, hacer una interfaz abstracta con funciones comunes de gestión de datos en el nivel físico. De ese modo, el gestor de almacenamiento puede intercambiarse, e incluso un mismo servidor MySQL puede utilizar diferentes motores de almacenamiento para diferentes bases de datos o para diferentes tablas en la misma base de datos. Esto permite utilizar el motor de almacenamiento más adecuado para cada necesidad concreta. También permite que terceros puedan implementar motores de almacenamiento nuevos para necesidades específicas, o adaptar el código de los existentes a ciertos requisitos de almacenamiento.

Los elementos que puede implementar un motor de almacenamiento son los siguientes:

Concurrencia. Es responsabilidad del motor implementar una política de bloqueos (o no implementar ninguna). Una estrategia de bloqueos por fila permite una mayor concurrencia, pero también consume más tiempo de procesamiento en aplicaciones en las que la concurrencia no es realmente grande.

Soporte de transacciones. No todas las aplicaciones necesitan soporte de transacciones.

Comprobación de la integridad referencial, declarada como restricciones en el DDL de SQL.

Almacenamiento físico, incluyendo todos los detalles de la representación en disco de la información.

Soporte de índices. Dado que la forma y tipo de los índices depende mucho de los detalles del almacenamiento físico, cada motor de almacenamiento proporciona sus propios métodos de indexación (aunque algunos como los árboles B casi siempre se utilizan).

Cachés de memoria. La eficiencia de los cachés de datos en memoria depende mucho de cómo procesan los datos las aplicaciones. MySQL implementa cachés comunes en el gestor de conexiones y la caché de consultas, pero algunos motores de almacenamiento pueden implementar cachés adicionales.

El gestor de conexiones

La gestión de conexiones es responsable de mantener las múltiples conexiones de los clientes. Un gestor de conexiones inexistente o laxo simplemente crearía una conexión por cada cliente conectado. No obstante, las conexiones consumen recursos de máquina, por lo tanto crearlas y destruirlas son también procesos costosos. Por eso, el gestor de conexiones de MySQL puede configurarse para limitar el número de conexiones concurrentes, y también implementar un pool de conexiones.

El problema es que muchas aplicaciones abren una conexión manteniéndola abierta y ociosa durante mucho tiempo (por ejemplo, durante toda la sesión de un

usuario, que de vez en cuando se levanta para diferentes tareas mundanas como tomar café), y solo “de vez en cuando” se utiliza un hilo de ejecución para realizar una consulta, que además, típicamente tarda unos milisegundos. No tiene sentido mantener una conexión ociosa para cada usuario. De aquí proviene la idea de los pools de conexiones: hay un número de conexiones disponibles, y cada vez que una aplicación hace una solicitud, se le asigna una conexión del pool que no esté ocupada. Lógicamente, la aplicación cliente no es consciente de este mecanismo.
En términos por ejemplo, de las interfaces JDBC, esto quiere decir que cada vez que se envía una sentencia con llamadas como *Statement.executeQuery()* realmente puede que se cree una nueva colección, o quizá se tome una del pool. Es decir, las llamadas a *Driver.getConnection()* y a *Connection.close()* no siempre determinan el tiempo de vida de una conexión real en MySQL.

Además de la reducción en el tiempo de establecimiento de conexión (si se rehúsa una conexión ya creada del pool), la técnica permite limitar el número de conexiones simultáneas. Dado que las conexiones consumen recursos, es mejor limitar este número que llevar a una carga excesiva en el servidor, que podría acabar en una caída del sistema o un comportamiento impredecible. Gracias a los pools de conexiones, puede darse servicio a muchos enlaces concurrentes con un número limitado de conexiones que se reutilizan.

El gestor de conexiones también se ocupa de la autentificación de los usuarios. La autentificación por defecto se basa en el nombre de usuario, la máquina desde la que se conecta y el *password*. El servidor puede también configurarse para soportar certificados X.509.

El procesamiento y optimizador de consultas

Cada vez que una consulta llega al gestor de MySQL, se analiza sintácticamente y se produce una representación intermedia de la misma. A partir de esa representación, MySQL toma una serie de decisiones, que pueden incluir el determinar el orden de lectura de las tablas, el uso de ciertos índices, o la re-escritura de la consulta en una forma más eficiente.

Existe la posibilidad de utilizar ciertas cláusulas en las consultas para ayudar al optimizador en su tarea, o bien podemos pedirle al servidor ciertas “explicaciones” sobre cómo ha planificado las consultas, para entender mejor su funcionamiento.
Dado que la optimización de las consultas depende de las capacidades del gestor de almacenamiento que se esté utilizando, el optimizador “pregunta” al gestor si

soporta ciertas características, y de este modo, puede decidir el tipo de optimización más adecuado.

La cache de consultas

MySQL implementa un caché de consultas, donde guarda las solicitudes y sus resultados enteros. De este modo, el procesador de consultas, antes de plantear la optimización, busca la consulta en la caché, para evitar realizar el trabajo en el caso de que tenga suerte y encuentre la consulta en la caché.

El control de concurrencia

El control de concurrencia en un gestor de bases de datos es el mecanismo que se utiliza para evitar que lecturas o escrituras simultáneas a la misma porción de datos terminen en inconsistencias o efectos no deseados. El mecanismo que se utiliza para controlar este acceso es el de los bloqueos (*locks*). La idea es, cada vez que una aplicación quiere acceder a una porción de los datos, se le proporciona un bloqueo sobre los mismos. Lógicamente, varias aplicaciones que quieran leer simultáneamente no tienen ningún problema en hacerlo, de modo que para la lectura se proporcionan bloqueos compartidos (*shared locks*). Sin embargo, varias actualizaciones o una actualización con consultas puede producir problemas. Por eso, para la escritura se proporcionan bloqueos exclusivos (*exclusive locks*).

Hay que tener en cuenta que la obtención y liberación de los bloqueos se realiza continuamente, y esto produce una sobrecarga en el procesamiento dentro del servidor. Además, hay diferentes políticas de bloqueo, por ejemplo, ¿es mejor bloquear cada tabla completa afectada o solo las filas de la tabla a las que quiere acceder una consulta? Dada la existencia de diferentes técnicas, el control de concurrencia en MySQL se divide entre el servidor y cada gestor de almacenamiento.

La gestión de transacciones y recuperación

La gestión de transacciones permite dotar de semántica “todo o nada” a una consulta o a un conjunto de consultas que se declaran como una sola transacción. Es decir, si hay algún problema y parte de la consulta o algunas de las consultas no consiguen llevarse a cabo, el servidor anulará el efecto parcial de la parte que

ya haya sido ejecutada. La recuperación permite “volver hacia atrás” (*rollback*) partes de una transacción.
Puede que una transacción implique más de una base de datos, y en ocasiones, a más de un servidor (transacciones distribuidas). MySQL proporciona soporte para todos estos tipos de transacciones, siempre que los gestores de almacenamiento utilizados en las bases de datos implicadas también lo soporten. [22]

2. ANALISIS Y DISEÑO DEL APLICATIVO INFORMATICO QUALITYSOFT

2.1. ELECCIÓN DE NORMA

Con la información recolectada se analizaron los estándares de calidad de software, se encontraron modelos viejos como son McCall fue el primero en ser presentado en 1977, y se originó motivado por US Air Force y DoD y el modelo de Boehm por Barry Boehm en 1978, los cuales fueron los antecesores de los modelos actualmente usados. En la Ingeniería de Software se necesitaba un modelo único para expresar la calidad de software, estos modelos han sido modificados y mejorados, una variante del modelo de McCall fue propuesta como estándar internacional para medición de calidad, el nombre formal fue “ISO 9126 Software Product Evaluation: Quality Characteristics and Guidelines for their Use”.

El cual se convirtió en el modelo más usado para la medición de calidad, tuvo su última revisión en el 2004 y actualmente es un modelo vigente. El último modelo es SQuaRE (Software Product Quality Requirements and Evaluation) o ISO/IEC 25000 nace con el objetivo de responder a las necesidades de los usuarios a través de un conjunto de documentos unificados cubriendo tres procesos de calidad complementarios: especificación de requisitos, medidas y evaluación. Por lo tanto, SQuaRE se creó a partir de la norma ISO/IEC 9126 y la norma ISO/IEC 14598, donde ellos (ISO/IEC 9126 y ISO/IEC 14598) pertenecen a la primera generación de estándares de calidad de un producto software. Por consiguiente, SQuaRE pertenece a la segunda generación de calidad de un producto software.
SQuaRE es una revisión de ISO/IEC 9126-1:2001, y conserva las mismas características de calidad de software.

Para la implementación de aplicativo informático QUALITYSOFT se selecciono la norma ISO/IEC 9126 porque es la más completa y conocida para la evaluación de producto de software, se descartó la utilización de la norma ISO/IEC 25000 o SQuaRE debido a que el momento de desarrollo del proyecto esta norma no estaba aprobada.

La base del aplicativo informático QUALITYSOFT sobre la cual las métricas son seleccionadas depende de los objetivos de negocio especificados para el producto y las necesidades del evaluador. Para ello utilizamos ISO/IEC 9126, especificando

el criterio de adaptación al modelo de calidad y las métricas seleccionadas, que serán evaluadas.

El procedimiento seguido para la obtención de un valor que integre las subcaracterísticas de cada factor “funcionalidad”, “confiabilidad”, “eficiencia”, “mantenimiento”, “portabilidad” y “usabilidad”, se fundamenta en la visión de usuarios, desarrolladores y gestores, sobre las métricas de calidad de software a aplicar. Para ello se han aplicado las siguientes estrategias, métodos, procedimientos y plantillas:

- Lista de Chequeo informada por usuarios, desarrolladores y gestores, expresando su valoración/peso de los factores de calidad. Los resultados se consolidan a nivel global indicando el peso de cada uno de los parámetros en el valor final.
- Propuesta de varias métricas por cada una de las subcaracterísticas y agregación a nivel global para obtener un valor conjunto del factor, cubriendo las subcaracterísticas.
- Aplicación de plantillas estándar a las métricas seleccionadas, para su definición, asignación de valores límite y su desglose en categorías.
- La utilización de las plantillas sobre los datos registrados, permite obtener: valores límite de cada una de las métricas, criterios de evaluación por categoría y su agregación para obtener una valoración a nivel de subcaracterística y característica.
- Las métricas de niveles de agregación superiores se calculan a partir de los valores de métricas del nivel de agregación inferior. El nivel de agregación superior está formado por los factores de calidad, definidos, que constituyen las características y subcaracterísticas seleccionadas de la norma ISO/IEC 9126.
-

Las métricas se validan de forma empírica mediante una relación causa-efecto y mediante la extracción de conclusiones.

2.2. RUP (RATIONAL UNIFIED PROCESS)

Teniendo en cuenta la investigación y el aplicativo informático a desarrollarse contó con las siguientes fases:

Inicio: En esta etapa se realizó la investigación de los estándares de calidad, seleccionados mediante la comparación de los mismos bajo criterios objetivos para determinar las métricas adecuadas que se usan en el proceso de auditoría de la calidad de software y así poder establecer las escalas de medición.
Además se estableció los requisitos iníciales del producto a desarrollar.

Elaboración: En esta etapa se inició el análisis y diseño del aplicativo basados en los requisitos iníciales adquiridos anteriormente y ayudados de los diagramas UML que permitió tener una visión clara del software a desarrollar.
Mediante los diagramas se permite hacer correcciones o incluir nuevos requisitos para el prototipo a implementar.

Construcción: El objetivo de esta etapa fue implementar el paquete con los cambios en los requerimientos, análisis y diseño que se obtuvo una vez realizadas las respectivas pruebas de funcionamiento del aplicativo.

Transición: En esta etapa se obtiene el paquete informático con las funcionalidades básicas y listas para ser usado en el proceso de auditoría. Además se elaboró la documentación completa de QUALITYSOFT.

Tabla 13. Actividades planificadas

ACTIVIDADES LOGICAS ANTERIORES	ACTIVIDADES PLANIFICADAS			ACTIVIDADES LOGICAS POSTERIORES
	ORDEN	DETALLE	DURACIÓN EN SEMANAS	
-	A	Investigación de los estándares de calidad	4	B
A	B	Selección de los estándares de calidad a usar	4	C
B	C	Análisis y comparación de las métricas de calidad de los estándares de calidad seleccionados	10	D,I
C	D	Diseño e implementación del aplicativo	4	E,F
D	E	Pruebas del aplicativo	1	F

D	F	Optimización de código	1	G,I
F	G	Pruebas del aplicativo a las instituciones universitarias	2	H
G	H	Comparación de resultados	2	I
C,F,H	I	Documentación	32	-

2.3. ANALISIS DEL APLICATIVO INFORMATICO QUALITYSOFT

En el análisis de este proyecto se realizó la obtención de requerimientos para el aplicativo QUALITYSOFT.

2.3.1. Listado de requerimientos. Para la construcción del aplicativo informático se diferenciaron tres módulos principales: El Módulo de Administración de la Aplicación: tiene como fin el registro y evaluación de proyectos, así mismo permite la visualización de proyectos ya realizados y hacer un análisis de esos datos, también permite agregar y eliminar preguntas, como modificar perfiles de usuario.

El Módulo Crear Proyecto: el cual permite ingresar los datos del proyecto de auditoría, seleccionar los factores y evaluar las métricas que aparecen de cada factor seleccionado.

El Módulo Abrir Proyectos: muestra todos los proyectos que se han realizado y facilita la visualización de reportes, borrar el proyecto seleccionado y continuar desarrollando proyectos incompletos. Además permite hacer copias de seguridad de proyectos y restaurarlos.

La interfaz gráfica del mismo admite una interacción amigable entre el usuario y el ambiente, facilitando su manejo.

2.3.2. Objetivos

Código:	OBJ-001
Descripción:	
Desarrollar una herramienta informática que permita evaluar la calidad del software en el proceso de auditoría de sistemas, basado en la aplicación de las métricas seleccionadas.	
Estado:	CRITICO
Comentarios:	

Código:	OBJ-002
Descripción:	
Implementar la herramienta informática que ayude a realizar el proceso de auditoría a la calidad del software y realizar las pruebas en los productos de software desarrollados en una de las instituciones de educación superior.	
Estado:	CRITICO
Comentarios:	

2.3.3. Requisitos iníciales

2.3.3.1. Requerimientos funcionales

Código:	RF- 001
Objetivo asociado:	OBJ-001, OBJ-002
Descripción:	
El sistema debe garantizar el acceso al sistema validado y verificado	
Estado:	APROBADO
Comentarios:	Los posibles tipos de usuario son: auditor y administrador, cada usuario debe tener un Nombre de usuario y una contraseña para el acceso.

Código:	RF- 002
Objetivo asociado:	OBJ-001, OBJ-002
Descripción:	
El sistema debe permitir administrar nuevos usuarios como son auditor y administrador	

Estado:	APROBADO
Comentarios:	Los posibles tipos de usuario son: administrador El administrador registra nuevo auditor. Solo el administrador elimina, consulta y modifica el perfil de usuarios. El Administrador tiene su nombre y clave para acceso al sistema.

Código:	RF- 003
Objetivo asociado:	OBJ-001, OBJ-002
Descripción:	
El sistema debe permitir administrar preguntas	
Estado:	APROBADO
Comentarios:	Los posibles tipos de usuario son: administrador El administrador puede agregar, eliminar, consultar preguntas El Administrador tiene su nombre y clave para acceso al sistema

Código:	RF- 004
Objetivo asociado:	OBJ-001, OBJ-002
Descripción:	
El sistema debe administrar proyectos	
Estado:	APROBADO
Comentarios:	Los posibles tipos de usuario son: auditor, administrador El auditor o administrador puede crear nuevo proyecto, abrir y visualizar un reporte de cada proyecto realizado, y continuar desarrollando proyectos incompletos. El administrador puede borrar proyectos del sistema, creando copias de seguridad. El auditor o administrador tiene su nombre y clave para acceso al sistema

Código:	RF- 005
Objetivo asociado:	OBJ-001, OBJ-002
Descripción:	
El sistema debe generar una lista de factores para analizar	
Estado:	APROBADO
Comentarios:	Los posibles tipos de usuario son: auditor, administrador El auditor o administrador puede seleccionar factores para ser analizados y evaluados en algún proyecto. El auditor o administrador debe asignar porcentajes cuando selecciona algún factor.

Código:	RF- 006
Objetivo asociado:	OBJ-001, OBJ-002
Descripción:	
El sistema debe mostrar las métricas, preguntas	
Estado:	APROBADO
Comentarios:	Los posibles tipos de usuario son: auditor, administrador El auditor o administrador puede evaluar las métricas o preguntas de cada factor seleccionado. El auditor o administrador puede hacer observaciones y no conformidades de cada métrica o pregunta

2.3.3.2. Requerimientos no funcionales

Código:	RNF – 001
Objetivo asociado:	OBJ – 001, OBJ – 002
Descripción:	
El sistema debe tener interfaz totalmente gráfica, fácil e intuitiva de manejar, que permita y facilite la Auditoría de sistemas.	

Código:	RNF – 002
Objetivo asociado:	OBJ – 001, OBJ – 002, OBJ -003, OBJ -004, OBJ -005
Descripción:	
El sistema debe estar implementado en el lenguaje de programación orientado a objetos C#.	
Código:	RNF – 003
Objetivo asociado:	OBJ – 001, OBJ – 002, OBJ -003, OBJ -004, OBJ -005
Descripción:	
El sistema debe usar el gestor de base de datos MySql.	

2.3.3.3. Matriz de rastreabilidad

Tabla 14. Matriz de rastreabilidad

	OBJ-001	OBJ-002
RF-001	✓	✓
RF-002	✓	✓
RF-003	✓	✓
RF-004	✓	✓
RF-005	✓	✓
RF-006	✓	✓
RNF-001	✓	✓
RNF-002	✓	✓
RNF-003	✓	✓

2.4. DISEÑO DEL APLICATIVO INFORMATICO QUALITYSOFT

2.4.1. Requerimientos de casos de uso

Nombre	Administrar proyecto
Resumen	El Auditor podrá crear, consultar un proyecto, continuar un proyecto incompleto
Entradas	
Nombre, clave, Nombre de proyecto	
Resultados	
El sistema creará el proyecto.	

Nombre	Administrar proyecto
Resumen	El Administrador podrá crear o consultar un proyecto, continuar un proyecto incompleto, eliminar proyectos
Entradas	
Nombre, clave, Nombre de proyecto	
Resultados	
El sistema actualiza / creará el proyecto.	

Nombre	Administrar proyecto
Resumen	El Administrador podrá agregar, eliminar preguntas, modificar auditores, modificar administrador
Entradas	
Nombre, apellido, clave, teléfono, email, nombre de pregunta.	
Resultados	
El sistema se actualiza.	

Nombre	Procesar métricas
Resumen	El Auditor o Administrador podrá seleccionar los factores a evaluar, darles un porcentaje de prioridad y evaluar las métricas.
Entradas	
Listado de los factores, porcentajes.	
Resultados	
El sistema mostrará las métricas de los factores seleccionados.	

Nombre	Mostrar informes
Resumen	El Auditor o Administrador podrá seleccionar, mostrar e imprimir un proyecto
Entradas	
Nombre de proyecto, Fecha y Hora	
Resultados	
El sistema mostrará e imprimirá el proyecto.	

2.4.2. Listado de casos de uso

Identificador:	CU 1
Nombre de caso de uso:	**.Administrar proyecto**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá crear o consultar un proyecto, continuar desarrollando un proyecto incompleto**
Curso básico de eventos:	1. El usuario selecciona la opción crear un proyecto o consultar un proyecto, continuar un proyecto 2. El sistema permite la operación 3. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 1.1
Nombre de caso de uso:	**. Crear proyecto**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá crear el proyecto**
Curso básico de eventos:	1. El usuario selecciona la opción crear un proyecto 2. El sistema muestra los campos nombre proyecto, tipo de proyecto 3. El usuario digita los campos y acepta la operación 4. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	3.1. Si los datos son inválidos, el sistema muestra mensaje: “Error en validación” 3.2 El sistema elimina los campos para volver a comenzar
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	Ir al caso de uso 2.1 seleccionar factores

Identificador:	CU 1.2
Nombre de caso de uso:	**.Consultar proyecto**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá consultar el proyecto y continuar desarrollando un proyecto incompleto**
Curso básico de eventos:	1. El usuario selecciona abrir proyecto

	2. El sistema muestra un listado de proyectos 3. El usuario selecciona un proyecto mostrar y acepta la operación 4. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	3.2 Puede imprimir el reporte del proyecto seleccionado. 2.1 Podrá continuar desarrollando un proyecto incompleto.
Puntos de extensión:	3.1 El sistema muestra un listado de proyectos por nombre.
Pre-Condiciones:	El proyecto este creado.
Post-Condiciones:	

Identificador:	CU 2
Nombre de caso de uso:	**. Procesar métricas**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá seleccionar factores, evaluar métricas**
Curso básico de eventos:	1. El usuario selecciona los factores a analizar y evaluar las métricas 2. El sistema permite la operación 3. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de extensión:	

Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 2.1
Nombre de caso de uso:	**. Seleccionar factores**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá seleccionar factores**
Curso básico de eventos:	1. El sistema muestra un listado de factores 2. El usuario selecciona los factores a analizar y coloca porcentajes a cada factor seleccionado. 3. El sistema permite la operación 1. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	3.1 Si los porcentajes son inválidos, el sistema muestra mensaje: “Error en validación” 3.2 el sistema vuelve a intentar
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	Ir al caso de uso 2.2 evaluar métricas

Identificador:	CU 2.2
Nombre de caso de uso:	**. Evaluar métricas**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá evaluar las métricas relacionadas a cada factor**

Curso básico de eventos:	1. El sistema muestra las métricas de cada factor 2. El usuario evalúa cada métrica de los factores y podrá agregar observaciones y no conformidades por cada métrica. 3. El sistema permite la operación 4. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	3.1 Si los datos son inválidos, el sistema muestra mensaje: “Error en validación” 3.2 el sistema vuelve a intentar
Puntos de extensión:	2.2el usuario podrá agregar notas de observación de la métrica evaluada 2.3el usuario podrá agregar notas de no conformidades de la métrica evaluada 2.4el usuario podrá guardar el proyecto
Pre-Condiciones:	
Post-Condiciones:	Ir al caso de uso 3.1 evaluar métricas

Identificador:	CU 3
Nombre de caso de uso:	**.Mostrar Informes**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá mostrar informe**
Curso básico de eventos:	1. El usuario selecciona el proyecto y muestra un informe. 2. El sistema permite la operación 3. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos	

alternativos:	
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 3.1
Nombre de caso de uso:	**.Mostrar Informe**
Autor:	
Actores involucrados:	Auditor
Resumen:	**El usuario podrá visualizar informes de proyectos**
Curso básico de eventos:	1. El sistema muestra un listado de proyectos realizados 2. El usuario selecciona el proyecto y muestra un informe. 3. El sistema permite la operación 4. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de extensión:	2.2 puede imprimir el informe
Pre-Condiciones:	Exista el proyecto.
Post-Condiciones:	

Identificador:	CU 1.3
Nombre de caso de uso:	**.Preguntas**
Autor:	

Actores involucrados:	Administrador
Resumen:	**El administrador podrá agregar, consultar, eliminar una pregunta de la base de datos**
Curso básico de eventos:	4. El administrador selecciona la opción agregar, eliminar un pregunta 5. El sistema permite la operación 6. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 1.3.1
Nombre de caso de uso:	**.Agregar**
Autor:	
Actores involucrados:	Administrador
Resumen:	**El administrador podrá agregar una pregunta de la base de datos**
Curso básico de eventos:	1. El administrador selecciona la opción agregar 2. Administrador ingresa la respectiva pregunta 3. El sistema permite la operación 4. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de	

extensión:	
Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 1.3.2
Nombre de caso de uso:	**.Eliminar**
Autor:	
Actores involucrados:	Administrador
Resumen:	**El administrador podrá consultar, eliminar una pregunta de la base de datos**
Curso básico de eventos:	1. El administrador selecciona la opción eliminar una pregunta 2. El sistema muestra un listado de preguntas 3. El administrador selecciona la pregunta a eliminar 4. El sistema permite la operación 5. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 1.4.1
Nombre de caso de uso:	**.Modificar perfiles**
Autor:	
Actores	Administrador

involucrados:	
Resumen:	**El administrador podrá modificar perfiles de usuarios.**
Curso básico de eventos:	1. El administrador selecciona la opción agregar un nuevo auditor, o modificar auditor existente. 2. Administrador ingresa datos principales del usuario como son: nombre, apellido, teléfono, email y clave. 3. El sistema permite la operación 4. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 1.4.1
Nombre de caso de uso:	**.Modificar perfiles**
Autor:	
Actores involucrados:	Administrador
Resumen:	**El administrador podrá modificar perfiles de usuarios.**
Curso básico de eventos:	5. El administrador selecciona la opción agregar un nuevo auditor, o modificar auditor existente. 6. Administrador ingresa datos principales del usuario como son: nombre, apellido, teléfono, email y clave. 7. El sistema permite la operación 8. El sistema muestra un mensaje exitoso “Solicitud

	procesada”
Caminos alternativos:	
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	

Identificador:	CU 1.4.3
Nombre de caso de uso:	**.Abrir proyecto**
Autor:	
Actores involucrados:	Administrador
Resumen:	**El administrador podrá visualizar, consultar un proyecto.**
Curso básico de eventos:	1. El sistema muestra un listado de proyectos. 2. El administrador selecciona un proyecto para mostrar 3. El sistema permite la operación 4. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	1.1 El administrador puede continuar desarrollando un proyecto incompleto. ir a caso de uso 2.2 evaluar métricas
Puntos de extensión:	1.2 El administrador puede borrar proyecto. 1.3 El administrador puede hacer copia de seguridad de proyectos y restaurarlos.
Pre-Condiciones:	
Post-Condiciones:	1.4 ir a caso de uso 2.2 evaluar métricas

Identificador:	CU 1.4.2
Nombre de caso de uso:	**.Nuevo Proyecto**
Autor:	
Actores involucrados:	Administrador
Resumen:	**El administrador podrá crear nuevo proyecto.**
Curso básico de eventos:	1. El sistema le muestra un campo para ingresar nombre de proyecto. 2. El sistema permite la operación 3. El sistema muestra un mensaje exitoso “Solicitud procesada”
Caminos alternativos:	
Puntos de extensión:	
Pre-Condiciones:	
Post-Condiciones:	ir al caso de uso 2.1 seleccionar factores

2.4.3. Diagramas de casos de uso

Figura 25. Diagrama de casos de uso. general

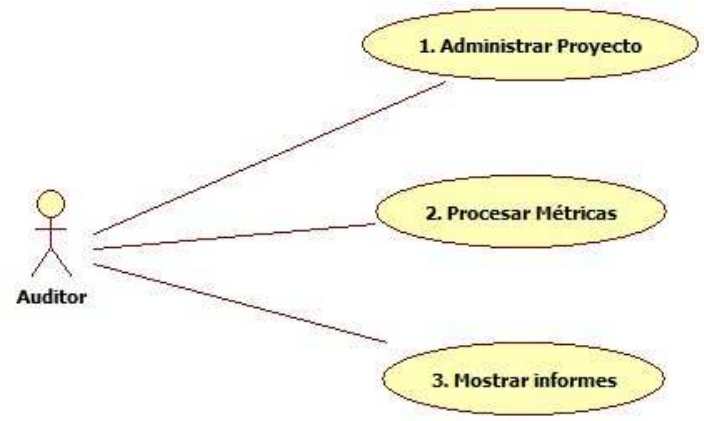


Figura 26. Diagrama de casos de uso. administrar proyecto

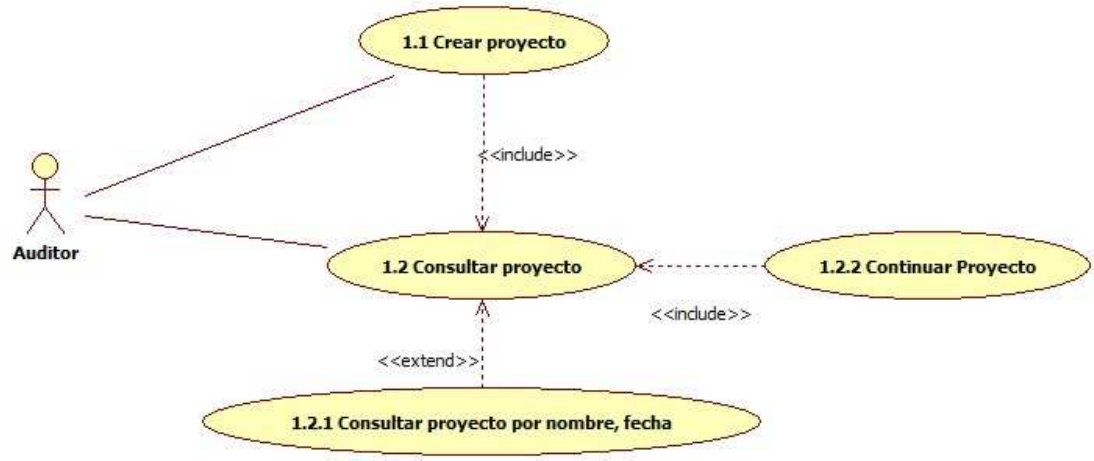


Figura 27. Diagrama de casos de uso. procesar métricas

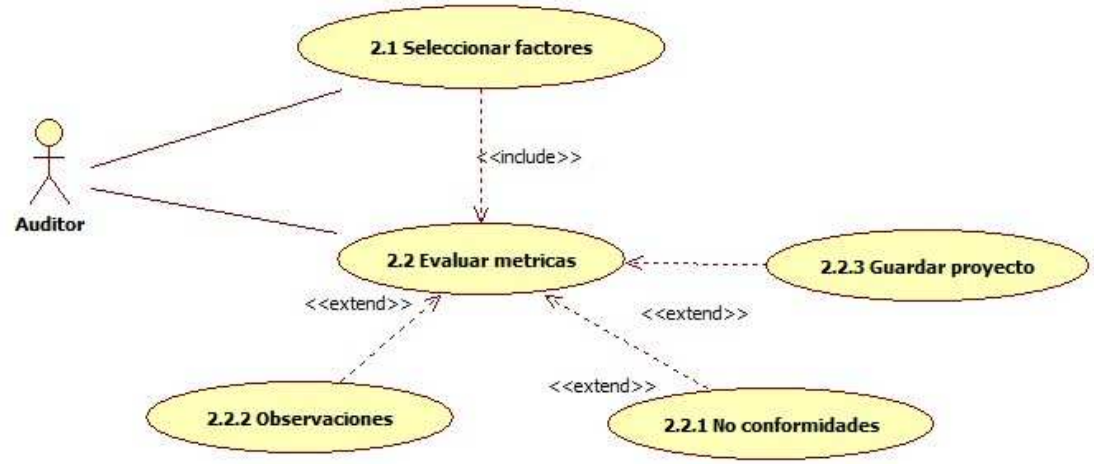


Figura 28. Diagrama de casos de uso. mostrar informe

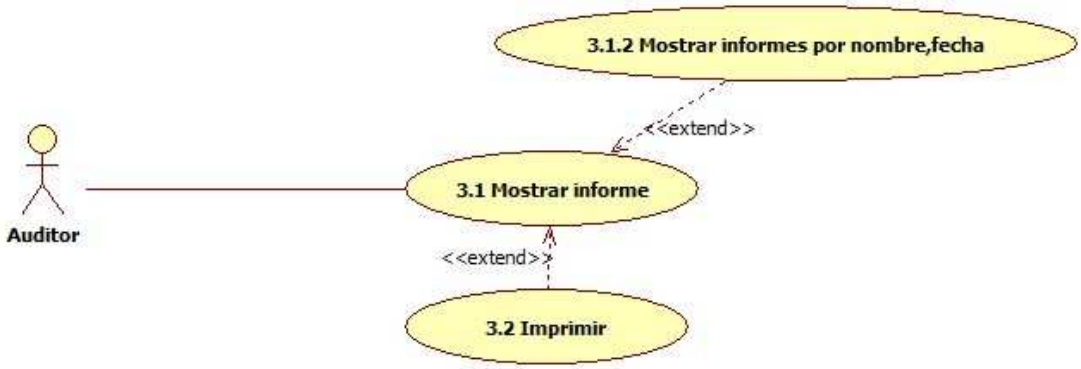


Figura 29. Diagrama de casos de uso. administrar proyecto (preguntas)

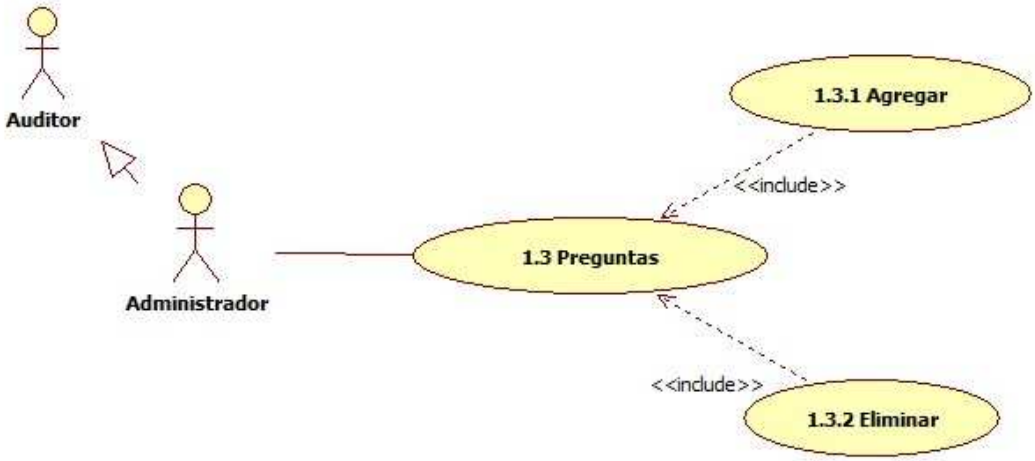
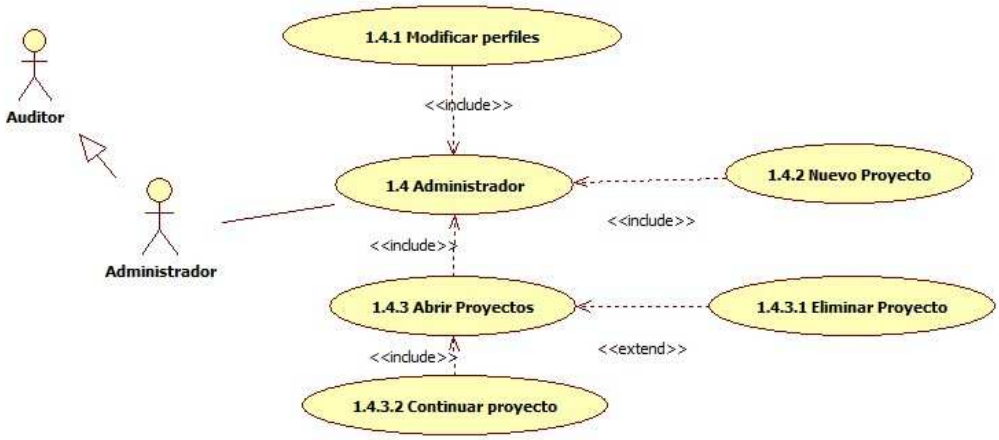


Figura 30. Diagrama de casos de uso. administrar proyecto (administrador)



2.4.4. Diseño de la base de datos

Figura 31. Diagrama de base de datos (esquema proyectos auditores-administrador)

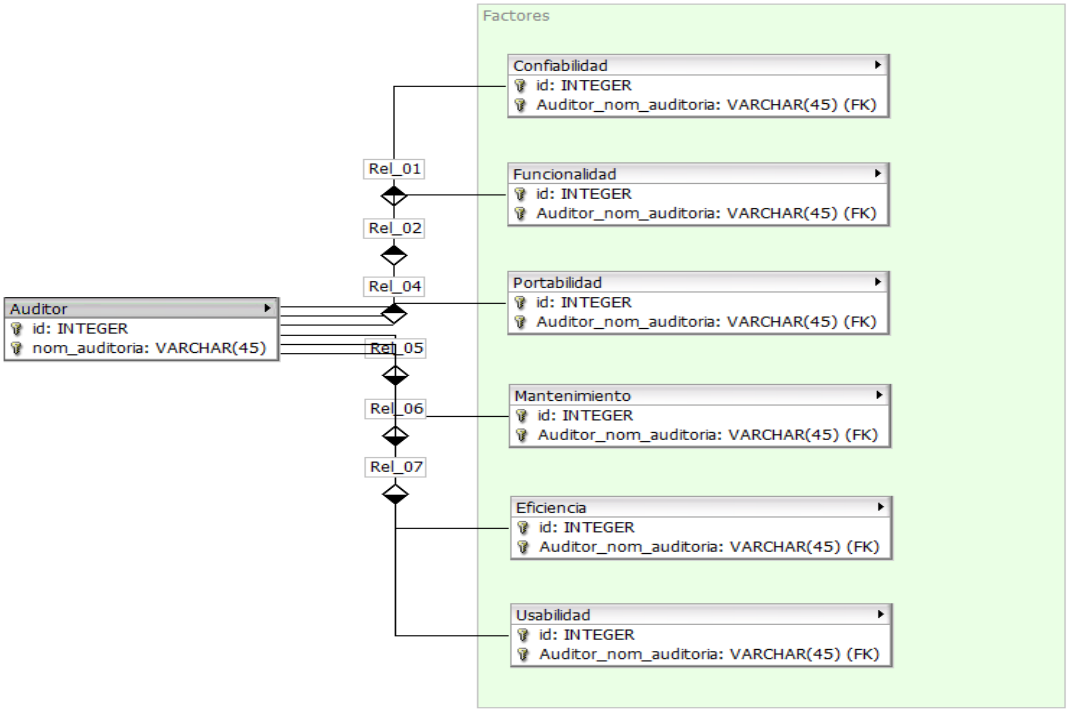
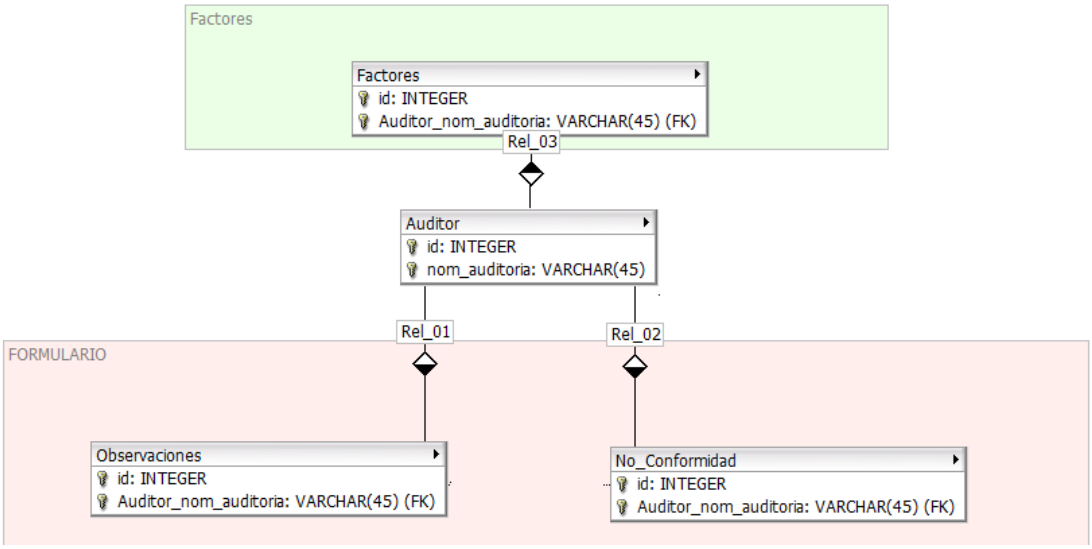


Figura 32. Diagrama de base de datos (esquema formatos observaciones-no conformidades)



2.4.5. Descripción tablas bases de datos

Tabla 15. Auditor

Auditor				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	Integer	4	si	No
cedula	integer	4	No	No
nom_Auditoría	varchar	100	No	No
nombres	varchar	20	No	No
apellidos	varchar	20	No	No
tipo_proyecto	varchar	100	No	No
teléfono	integer	4	No	No

Esta tabla representa la información de cada proyecto asociada a cada auditor

Tabla 16. Característica confiabilidad

Confiabilidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	Si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría)
Tipo	varchar	20	No	No
subcaracteristica	varchar	30	No	No
valor	float	4	No	No

Esta tabla representa la característica confiabilidad asociada a cada proyecto existente

Tabla 17. Característica usabilidad

Usabilidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
Tipo	Varchar	20	No	No
subcaracteristica	Varchar	30	No	No
Valor	Float	4	No	No

Esta tabla representa la característica Usabilidad asociada a cada proyecto existente

Tabla 18. Característica portabilidad

Portabilidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
Tipo	varchar	20	No	No
subcaracteristica	varchar	30	No	No
valor	float	4	No	No

Esta tabla representa la característica Portabilidad asociada a cada proyecto existente

Tabla 19. Característica eficiencia

Eficiencia				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
tipo	varchar	20	No	No
subcaracteristica	varchar	30	No	No
valor	Float	4	No	No

Esta tabla representa la característica Eficiencia asociada a cada proyecto existente

Tabla 20. Característica mantenimiento

Mantenimiento				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
tipo	varchar	20	No	No
subcaracteristica	varchar	30	No	No
valor	float	4	No	No

Esta tabla representa la característica mantenimiento asociada a cada proyecto existente

Tabla 21. Característica funcionalidad

Funcionalidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	Si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
tipo	varchar	20	No	No
subcaracteristica	varchar	30	No	No
Valor	float	4	No	No

Esta tabla representa la característica Funcionalidad asociada a cada proyecto existente

Tabla 22. Factores y porcentajes

Factores				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
nombre	varchar	20	No	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
porcentaje	integer	4	No	No
valida	integer	4	No	No

Esta tabla representa los factores evaluados y sus porcentajes asociados a cada proyecto existente

Tabla 23. Formato observaciones

Observaciones				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	Si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
No	integer	4	No	No
fecha	Varchar	20	No	No
caracteristica	varchar	20	No	No
subcaracteristica	varchar	30	No	No
tipo	varchar	10	No	No
nom_pregunta	Varchar	250	No	No
descripción	Text	Longitud +2 bytes	No	No

Esta tabla representa los formatos de observaciones asociados a cada proyecto existente

Tabla 24. Formato no conformidades

NoConformidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
nom_Auditoría	varchar	100	No	Auditor(nom_Auditoría
No	integer	4	No	No
fecha	Varchar	20	No	No
caracteristica	varchar	20	No	No
subcaracteristica	varchar	30	No	No
tipo	varchar	10	No	No
nom_pregunta	Varchar	250	No	No
Reporte_nc	text	Longitud +2 bytes	No	No
des_correctiva	text	Longitud +2 bytes	No	No

Esta tabla representa los formatos de no Conformidad asociados a cada proyecto existente

Tabla 25. Auditores

Auditores				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
cedula	integer	4	si	No
nombres	varchar	20	No	No
apellidos	varchar	20	No	No
email	varchar	30	No	No

telefono	integer	4	No	No
clave	varchar	20	No	No

Esta tabla representa los datos de los Auditores que se encuentran registrados en la B.D QUALITYSOFT que pueden realizar proyectos

Tabla 26. Administradores

Administradores				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
cedula	integer	4	si	No
nombres	varchar	20	No	No
apellidos	varchar	20	No	No
email	varchar	30	No	No
telefono	integer	4	No	No
Clave	varchar	20	No	No

Esta tabla representa los datos del Administrador que se encuentra registrado en la B.D QUALITYSOFT el cual se encarga de Agregar nuevos auditores a las B.D y también puede desempeñar el papel de un Auditor.

Tabla 27. adi_funcionalidad

adi_funcionalidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
id	integer	4	si	No

pregunta	text	Longitud +2 bytes	No	No
Si	bool	1	No	No
no	bool	1	No	No
no_aplica	bool	1	No	No
Obs	bool	1	No	No
no_con	bool	1	No	No

Esta tabla representa las preguntas añadidas por parte del Administrador en la característica Funcionalidad

Tabla 28. adi_confiabilidad

adi_confiabilidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
Pregunta	text	Longitud +2 bytes	No	No
si	bool	1	No	No
no	bool	1	No	No
no_aplica	bool	1	No	No
obs	bool	1	No	No
no_con	bool	1	No	No

Esta tabla representa las preguntas añadidas por parte del Administrador en la característica Confiabilidad

Tabla 29. adi_usabilidad

adi_usabilidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
pregunta	text	Longitud +2 bytes	No	No
si	bool	1	No	No
no	bool	1	No	No
no_aplica	bool	1	No	No
Obs	bool	1	No	No
no_con	bool	1	No	No

Esta tabla representa las preguntas añadidas por parte del Administrador en la característica Usabilidad

Tabla 30. adi_mantenimiento

adi_mantenimiento				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
id	integer	4	si	No
pregunta	text	Longitud +2 bytes	No	No
Si	bool	1	No	No
no	bool	1	No	No
no_aplica	bool	1	No	No

Obs	bool	1	No	No
no_con	bool	1	No	No

Esta tabla representa las preguntas añadidas por parte del Administrador en la característica Mantenimiento

Tabla 31. adi_portabilidad

adi_portabilidad				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
id	integer	4	Si	No
pregunta	text	Longitud +2 bytes	No	No
Si	bool	1	No	No
no	bool	1	No	No
no_aplica	bool	1	No	No
Obs	bool	1	No	No
no_con	bool	1	No	No

Esta tabla representa las preguntas añadidas por parte del Administrador en la característica Portabilidad

Tabla 32. adi_eficiencia

adi_eficiencia				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
Id	integer	4	si	No
Pregunta	text	Longitud +2 bytes	No	No

Si	bool	1	No	No
No	bool	1	No	No
no_aplica	bool	1	No	No
Obs	bool	1	No	No
no_con	bool	1	No	No

Esta tabla representa las preguntas añadidas por parte del Administrador en la característica Eficiencia

Tabla 33. log

Log				
NOMBRE CAMPO	**TIPO CAMPO**	**LONGITUD CAMPO**	**LLAVE PRIMARIA**	**LLAVE FORÁNEA**
nombre	Varchar	20	No	No
Fecha y hora	Varchar	30	No	No
acción	Varchar	50	No	No

Esta tabla representa el archivo de bitácora del aplicativo.

2.4.6. Diagramas de secuencia

Figura 33. Diagrama de secuencia (crear proyecto)

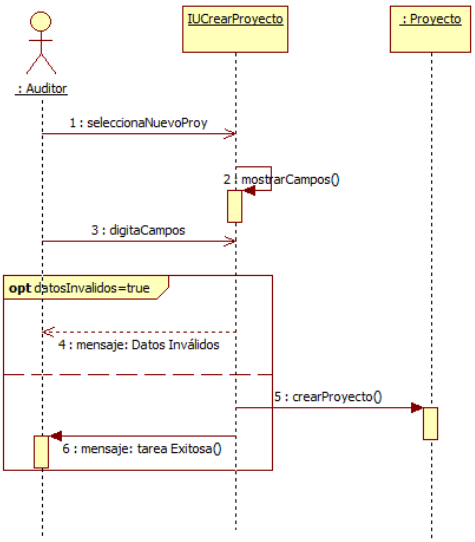


Figura 34. Diagrama de secuencia (seleccionar factores)

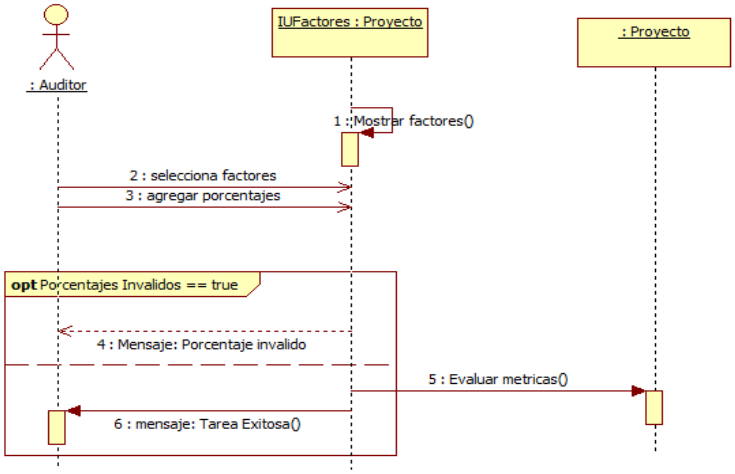


Figura 35. Diagrama de secuencia (mostrar informe)

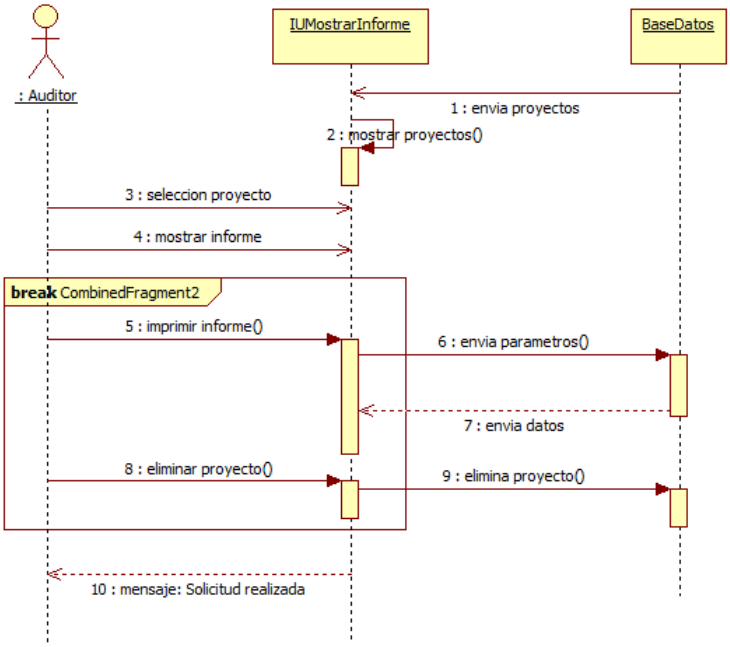
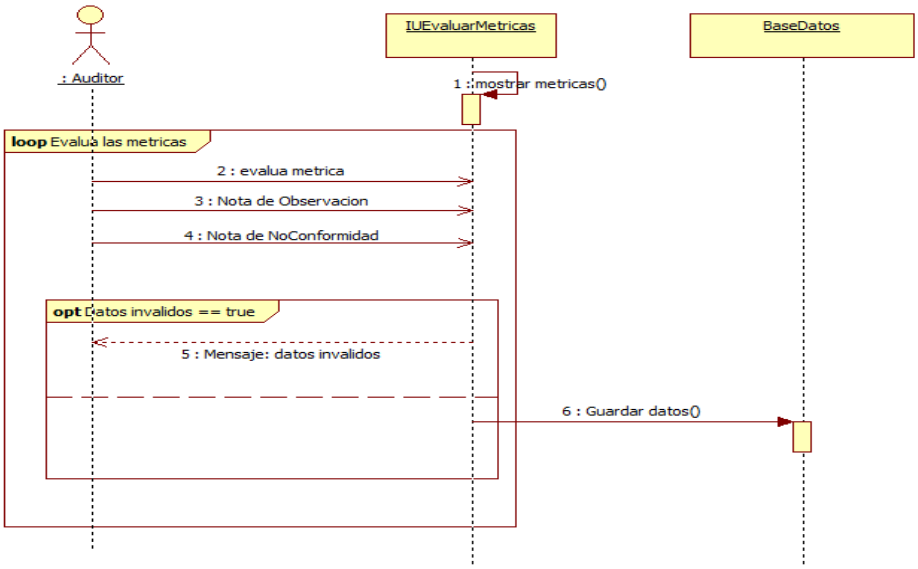
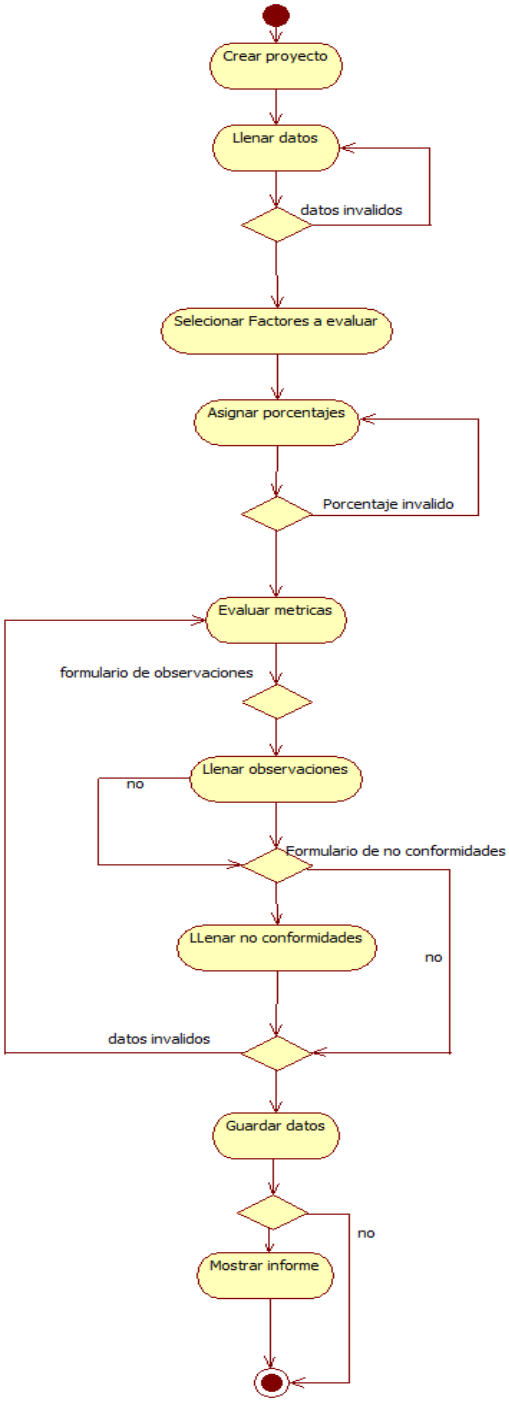


Figura 36. Diagrama de secuencia (evaluar metricas)



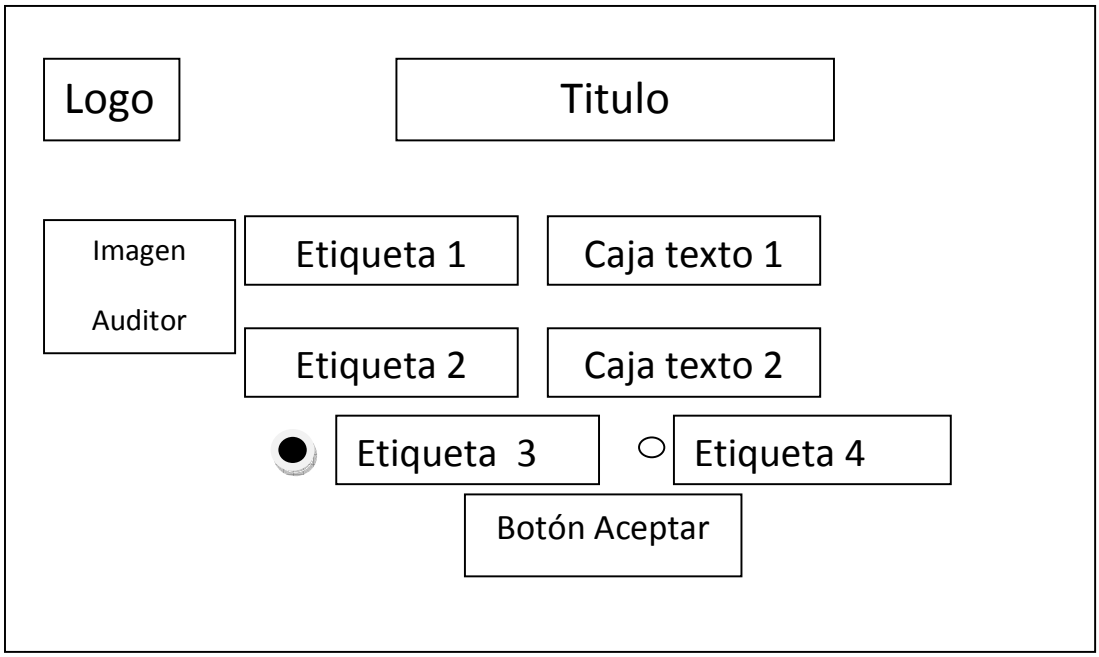
2.4.7. Diagrama de actividades

Figura 37. Diagrama de actividades



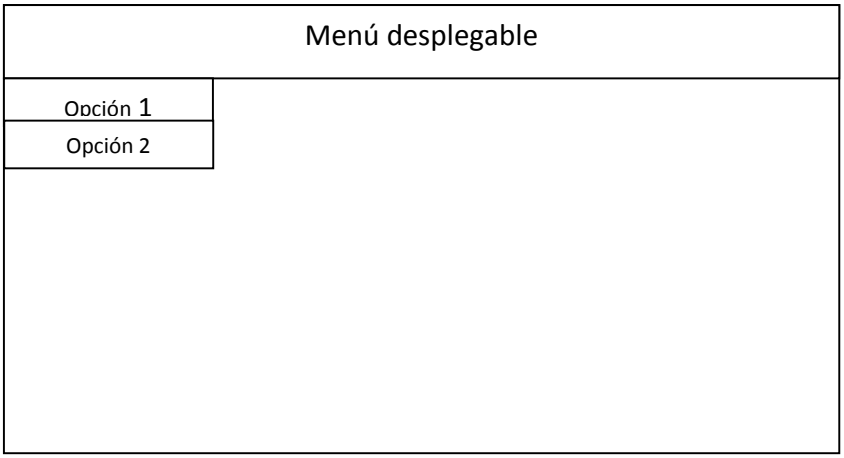
2.4.8. Diagrama de interfaces

Figura 38. Ventana de inicio - administrador



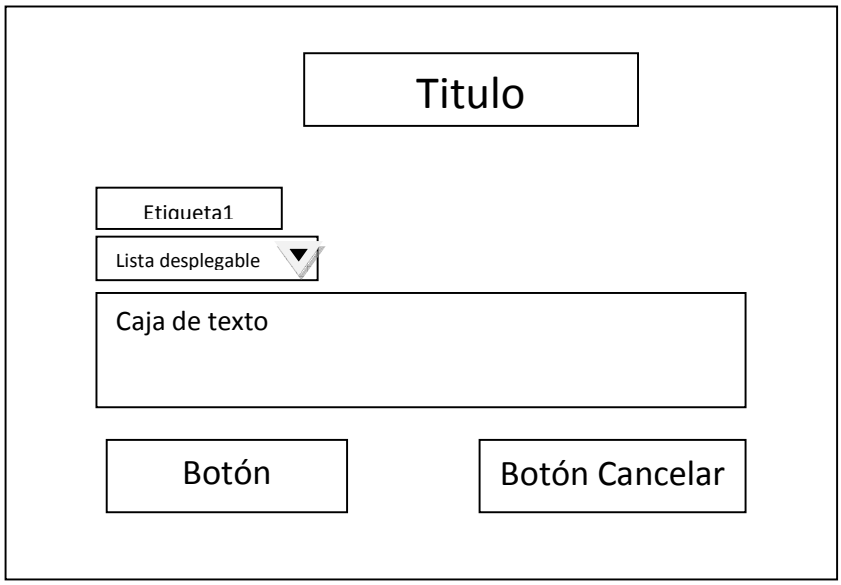
Logo: Aquí se ubicara el logo del programa
Titulo: Aquí estará el título “QUALITYSOFT”
Imagen auditor: Esta imagen representa al auditor y se carga cuando el *radio button* alado de la etiqueta 3 este marcado
Etiqueta 1: Tendrá el texto “identificador” y estará asociado a la caja de texto 1
Etiqueta 2: Tendrá el texto “clave” y estará asociado a la caja de texto 2
Etiqueta 3: Tendrá el texto “Auditor”
Etiqueta 4: Tendrá el texto “Administrador”
Caja texto1: En esta caja de texto se llenaran los datos del auditor (el identificador como aparece en la Base de Datos)
Caja texto2: En esta caja de texto se digitará la clave (para ser comprobada en la Base de Datos)
Botón Aceptar: Este botón se encarga de verificar que las cajas de texto estén llenas y con datos validos para pasar a la siguiente pantalla.

Figura 39. Menú: administrador



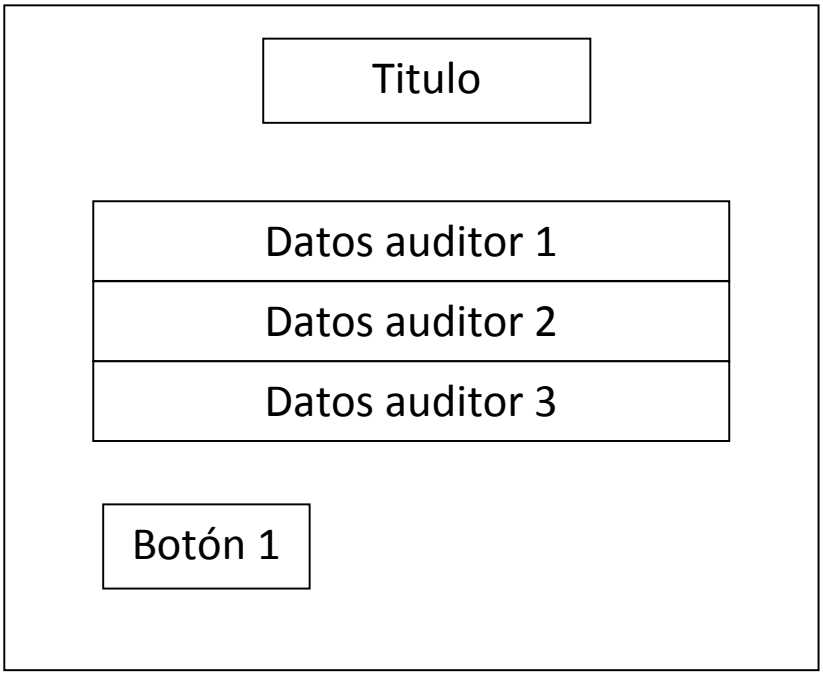
Menú desplegable: en este se encuentran las opciones: el de pregunta (donde se puede agregar nuevas métricas), administrador (donde se puede abrir proyectos crear proyectos y modificar el perfil del administrador), auditores (donde se puede modificar la información de los auditores registrados y agregar nuevos auditores)
Y la opción log(donde se muestra un registro de las actividades en el programa)
Opción: Muestra las opciones asociadas a cada menú.

Figura 40. Menú: pregunta – administrador



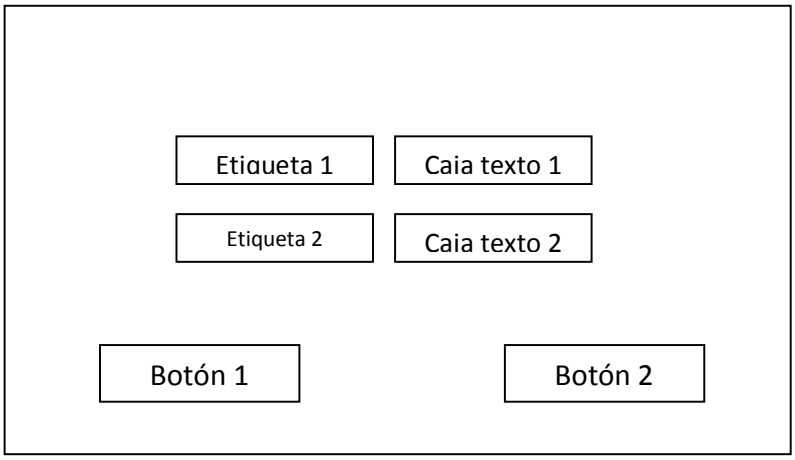
Titulo: Tendrá el título de “Nueva Pregunta”
Etiqueta 1: llevará el texto de “seleccioné categoría”
Lista desplegable: En esta se cargarán las categorías para que una nueva pregunta sea clasificada
Caja texto: Aquí se escribirá la pregunta que será agregada posteriormente
Botón Aceptar: Es el botón confirmar, el cual agregará la nueva pregunta
Botón Cancelar: Es el botón para cancelar la operación

Figura 41. Menú: auditores – administrador



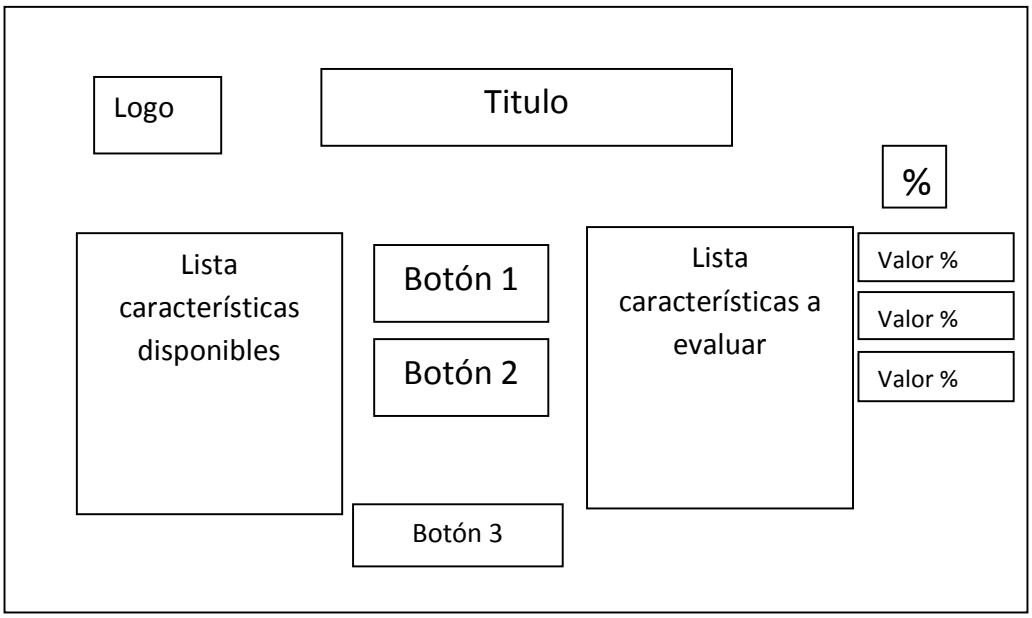
Titulo: llevará el texto “Auditores”
Datos Auditor: se cargarán todos los datos de los auditores para ser editados o agregar nuevos auditores
Botón: Este es el botón “confirmar para guardar los cambios”

Figura 42. Crear un nuevo proyecto – administrador



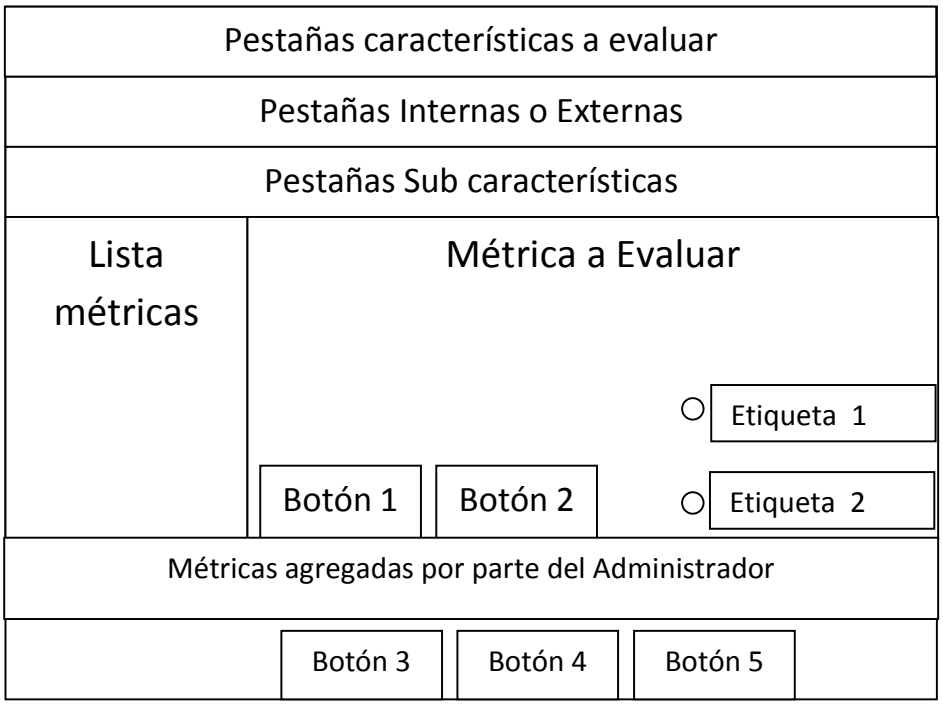
Etiqueta 1: Tendrá el texto “Tipo software”
Etiqueta 2: Tendrá el texto “Nombre proyecto”
Caja texto 1: Esta caja de texto está asociada a la etiqueta 1
Caja texto 2: Esta caja de texto está asociada a la etiqueta 2
Botón 1: Guarda los datos de las cajas de texto en la base de datos y permite pasar a la siguiente ventana
Botón 2: Este es el botón cancelar

Figura 43. Seleccionar factores



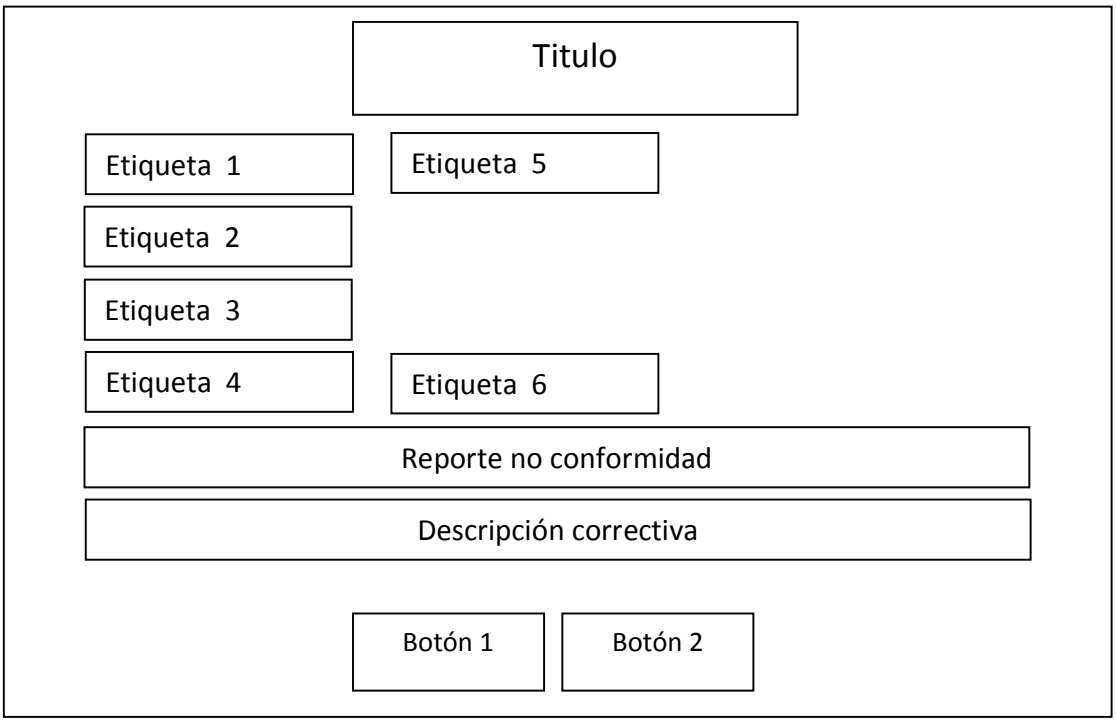
Logo: Aquí se ubicará el logo del programa
Titulo: Aquí estará el título “QUALITYSOFT”
Lista características disponibles: Muestra las 6 características disponibles para evaluar
Lista características a evaluar: Muestra las características que el usuario escogió para evaluar
Botón 1: Permite pasar la característica de la lista características disponibles a la lista características a evaluar
Botón 2: Permite pasar la característica de la lista características a evaluar a la lista características disponibles
Botón 3: Es el botón que verifica los datos en “valor” estén bien y la suma de esos valores sea 100, cuando esté correcto, permite pasar a la siguiente ventana
%: muestra el símbolo de “%”
Valor: Aquí se llenan los valores de los porcentajes que tendrá cada característica en la lista características a evaluar

Figura 44. Menú factores y métricas



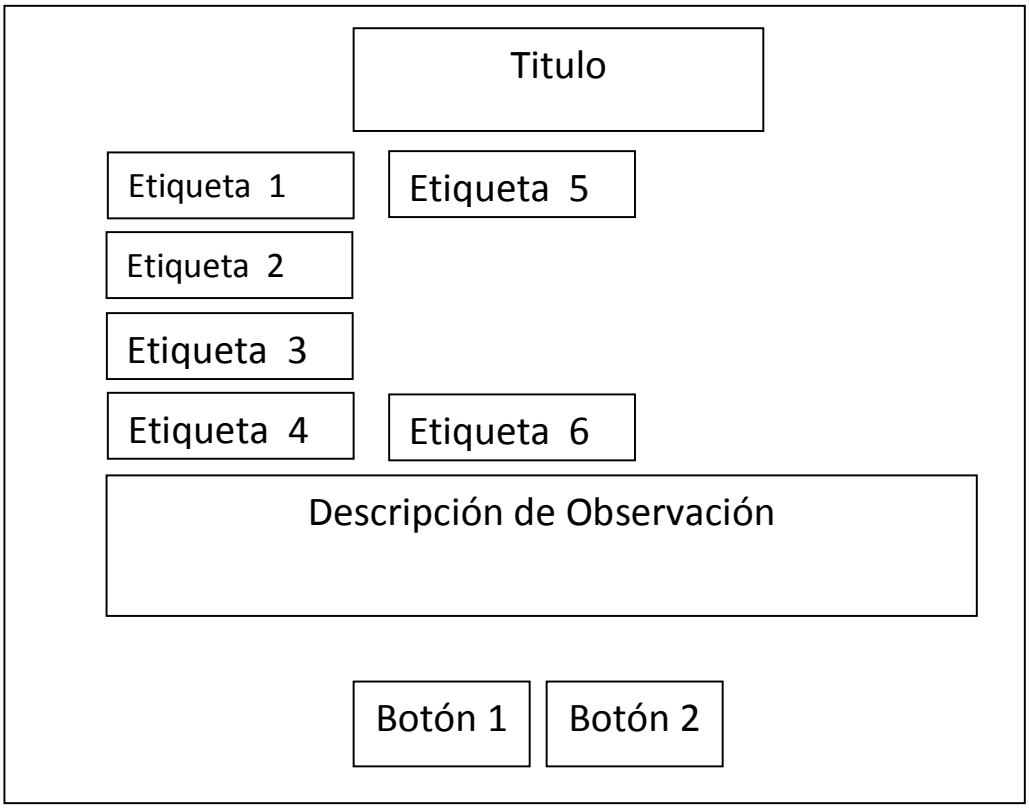
Pestaña características a evaluar: Muestra las características que se pueden evaluar en el proceso de Auditoría
Pestaña interna o externa: Muestra si se evalúa las características internas o externas
Pestaña sub características: Muestra las sub características que se pueden evaluar en el proceso de Auditoría
Lista métricas: Muestra las métricas a evaluar por cada sub característica
Métrica a evaluar: Muestra la métrica
Botón 1: Es el botón aceptar la métrica a evaluar
Botón 2: Es el botón de no aplica la métrica a evaluar
Botón 3: Es el botón finalizar de evaluar la característica
Botón 4: Muestra reportes parciales de los factores.
Botón 5: Es el botón restaurar los valores evaluar una característica desde" 0"
Etiqueta 1: Muestra el texto "observaciones" asociado a un *check box* para llamar al formato de observaciones
Etiqueta 2: Muestra el texto "no conformidades "asociado a un *check box* para llamar al formato de no conformidades
Métricas agregadas por administrador: En esta sección se cargan las preguntas agregadas por el administrador

Figura 45. Hoja de observaciones



Titulo: Lleva el texto “Hoja de No conformidad”
Etiqueta 1: Se carga la fecha
Etiqueta 2: Se carga la característica
Etiqueta 3: Se carga la sub característica
Etiqueta 4: Se carga el nombre de la pregunta
Etiqueta 5: Se carga el número de hoja de no conformidad
Etiqueta 6: Se carga el tipo si es interna o externa
Reporte con conformidad: campo para llenar la no conformidad
Descripción correctiva: campo para llenar la descripción a seguir
Botón 1: botón Aceptar y guarda en la base de datos
Botón 2: botón cancelar

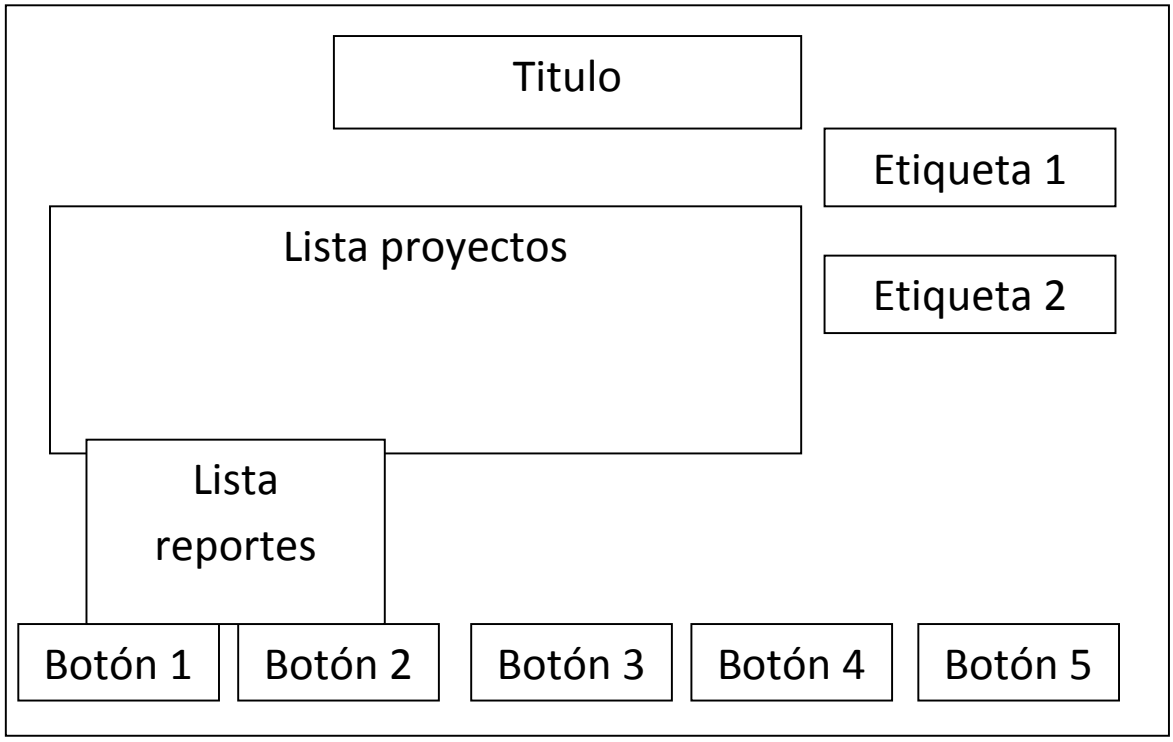
Figura 46 Hoja de no conformidades



Titulo: Lleva el texto “Hoja de Observaciones”
Etiqueta 1: Se carga la fecha
Etiqueta 2: Se carga la característica
Etiqueta 3: Se carga la sub característica
Etiqueta 4: Se carga el nombre de la pregunta

Etiqueta 5: Se carga el número de hoja de observaciones
Etiqueta 6: Se carga el tipo si es interna o externa
Descripción observación: campo para llenar la observación
Botón 1: botón Aceptar y guarda en la base de datos
Botón 2: botón cancelar

Figura 47. Abrir proyecto – administrador



Titulo: Lleva el texto “Proyectos”
Lista proyectos: se cargan los proyectos existentes en la base de datos
Etiqueta 1: Lleva el texto “Proyectos terminados”
Etiqueta 2: Lleva el texto “Proyectos inconclusos”
Lista reportes: Muestra los posibles tipos de reportes que se pueden mostrar
Botón 1: Botón para mostrar los reportes
Botón 2: Botón para continuar con los proyectos inconclusos
Botón 3: Botón para hacer una copia de seguridad de la base de datos
Botón 4: Botón para restaurar una base de datos
Botón 5: Botón cerrar proyecto

Figura 48. Ventana de inicio - auditor

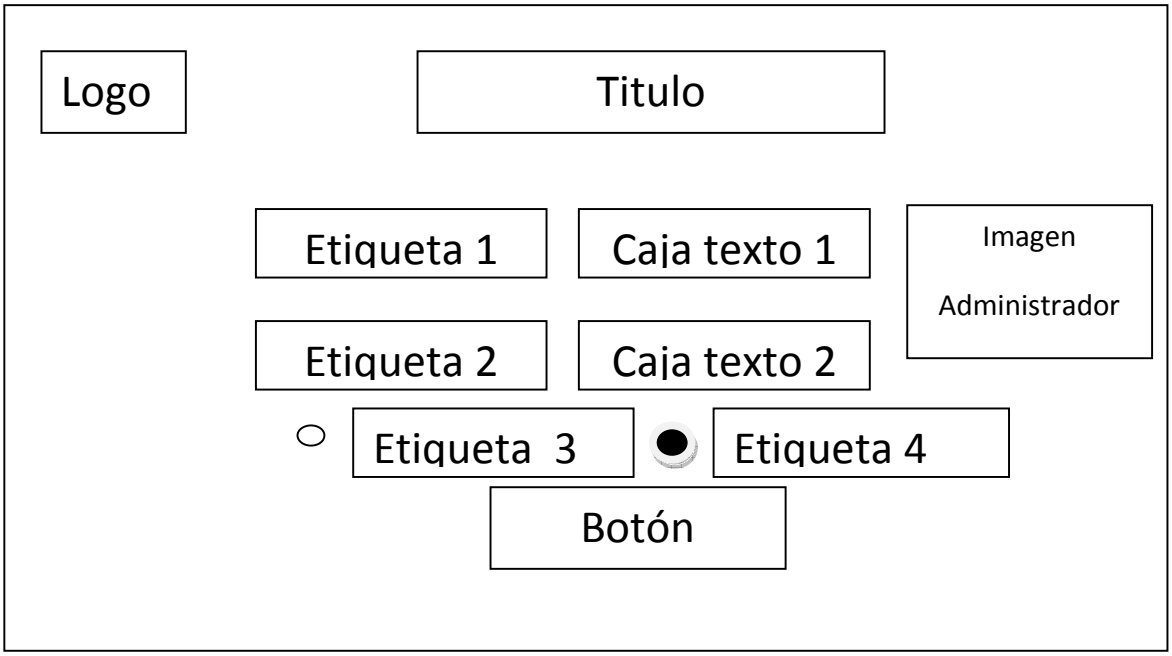
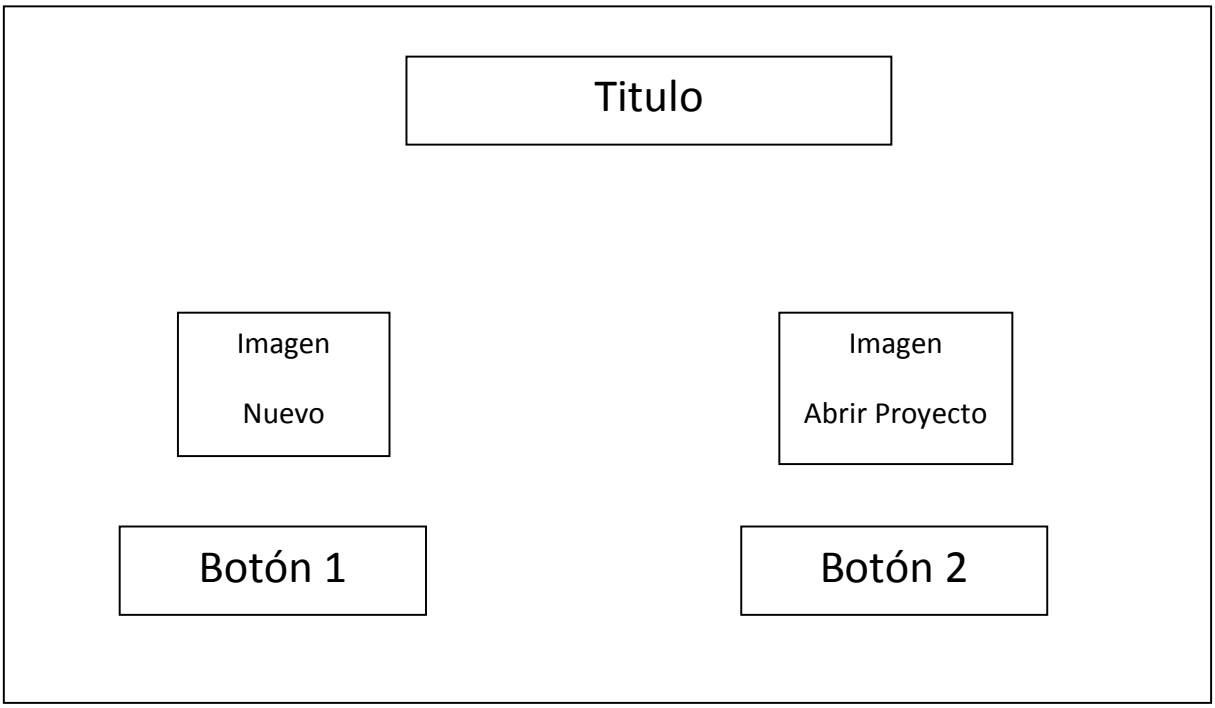


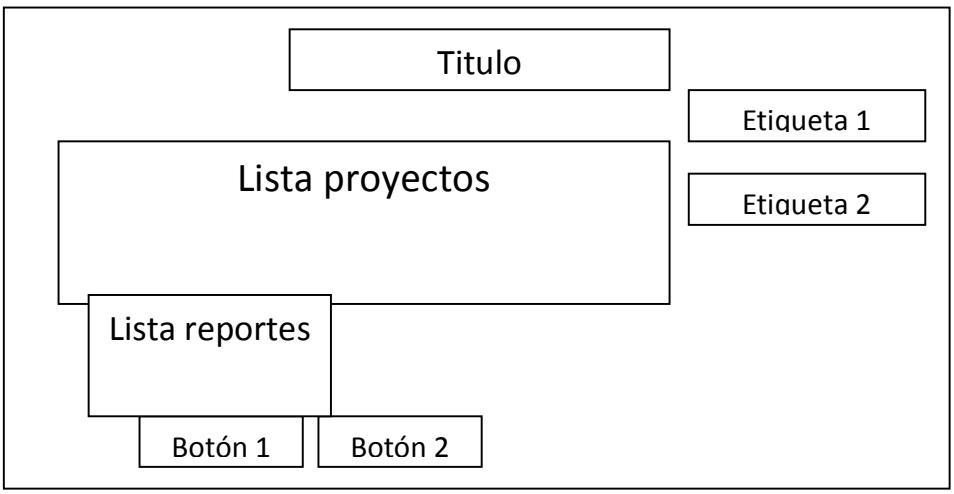
Imagen administrador: Esta imagen representa al administrador y se carga cuando el *radio button* al lado de la etiqueta 4 este marcado
Etiqueta 1: Tendrá el texto “identificador” y estará asociado a la caja de texto 1
Etiqueta 2: Tendrá el texto “clave” y estará asociado a la caja de texto 2
Etiqueta 3: Tendrá el texto “Auditor”
Etiqueta 4: Tendrá el texto “Administrador”
Caja texto1: En esta caja de texto se llenarán los datos del administrador (el identificador como aparece en la Base de Datos)
Caja texto2: En esta caja de texto se digitará la clave (para ser comprobada en la Base de Datos)
Botón: Este botón se encarga de verificar que las cajas de texto estén llenas y con datos validos para pasar a la siguiente pantalla.

Figura 49. Menú – auditor



Titulo: Tendrá el texto “QUALITYSOFT”
Imagen nuevo proyecto: Esta imagen representa la creación de un nuevo proyecto
Imagen abrir proyecto: Esta imagen representa el abrir un nuevo proyecto
Botón 1: Botón para crear nuevo proyecto
Botón 2: Botón para abrir nuevo proyecto

Figura 50. Abrir proyecto - auditor



Titulo: Lleva el texto “Proyectos”
Lista proyectos: se cargan los proyectos existentes en la base de datos
Etiqueta 1: Lleva el texto “Proyectos terminados”
Etiqueta 2: Lleva el texto “Proyectos inconclusos”
Lista reportes: Muestra los posibles tipos de reportes que se pueden mostrar
Botón 1: Botón para mostrar los reportes
Botón 2: Botón para continuar con los proyectos inconclusos

2.5. IMPLEMENTACION

2.5.1. Introducción

Para la implementación del aplicativo informático QUALITYSOFT se utilizó el siguiente software:

- Sistema Operativo: Microsoft® Windows® 7 ULTIMATE.
- Lenguaje de Programación: C-Sharp
- Plataforma de Desarrollo: Visual Studio 2010.

2.5.2 .Arquitectura QUALITYSOFT

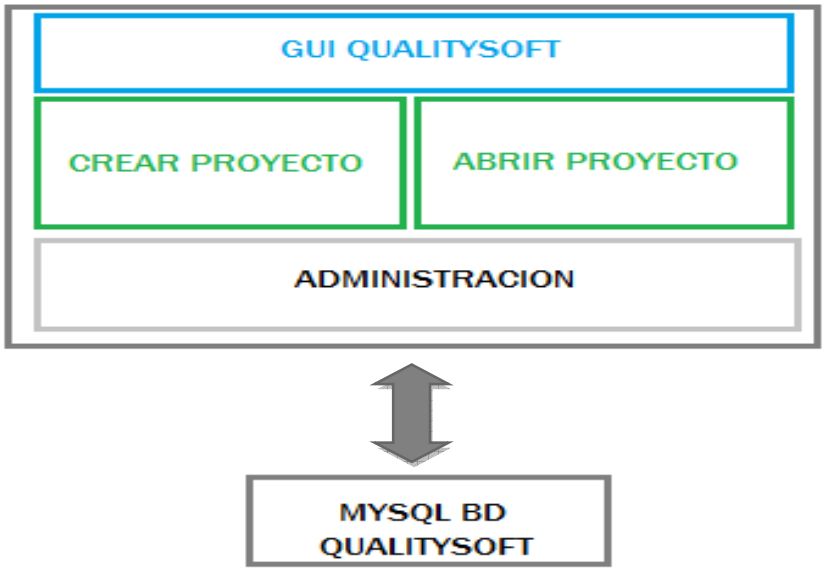
QUALITYSOFT lógicamente se compone de 3 módulos:

El Módulo de Administración de la Aplicación: que tiene como fin el registro y evaluación de proyectos, así mismo permite la visualización de proyectos ya realizados y hacer un análisis de esos datos, también admite agregar y eliminar preguntas y modificar perfiles de usuario.

El módulo crear nuevo proyecto: el cual solicita el nombre de la Auditoría para seleccionar los factores y evaluar las métricas que aparecen de cada factor seleccionado.

El módulo abrir proyectos: muestra todos los proyectos que se han realizado y facilita la visualización de reportes, también borrar el proyecto seleccionado o seguir desarrollando un proyecto incompleto y crear copias de seguridad de proyectos.

Figura 51. Arquitectura QUALITYSOFT



3. PRUEBAS Y RESULTADOS

3.1. ANALISIS DE FUNCIONALIDAD

El análisis de funcionalidad de la herramienta se desarrollo utilizando un computador INTEL CORE2QUAD con procesador 2,4GHZ, 3GB de RAM, Disco Duro de 250 GB, Sistema Operativo Microsoft Windows 7 ULTIMATE.
Para la realización de las pruebas con la herramienta QUALITYSOFT se hizo uso del aplicativo web KNOWGER: “AMBIENTE VIRTUAL PARA LA GESTIÓN DE CONOCIMIENTO PARA INTEGRAR LA LABOR ACADÉMICA E INVESTIGATIVA DE LA FACULTAD DE INGENIERIA” de la Universidad de Nariño.

3.2 .REQUISITOS PRUEBA GENERAL

Inicialmente deben estar en la base de datos QUALITYSOFT los datos personales del administrador.
Para la prueba se tendrá la Base de Datos vacía, y se procede a evaluar el aplicativo web KNOWER por medio de la sesión de estudiante.

3.3. PRUEBA GENERAL

Métricas Externas

Tabla 34. Usabilidad

Característica	Valor Obtenido	%
USABILIDAD	11,4933	16
Comprensibilidad	0,714286	bueno
Aprendizaje	0,8733	excelente
Operabilidad	0,7	bueno
Atractivo	0	deficiente
		FIN

Analizando este factor de usabilidad, se puede ver como la comprensibilidad en el aplicativo web evaluado es bueno porque los usuarios entienden el software, no tienen dificultades para la selección de las funciones que desean usar, es adecuado y puede ser usado para tareas particulares.

La subcaracterística aprendizaje es excelente, al usuario no le toma mucho tiempo entender y aprender a usar una nueva función lo cual hace que el tiempo que emplea para ir de una función a otra es muy corto.
En cuanto a operabilidad, se nota que el programa fue analizado en una etapa en la que se ha corregido previamente los errores y se han hecho las validaciones correspondientes

Los usuarios fácilmente aprenden a utilizar funciones específicas, y la eficacia de los sistemas de ayuda y documentación con los que cuenta el aplicativo web hace que los usuarios puedan operar y controlar el software.
En cuanto a atractivo, el aspecto del software es limitado ya que no se puede modificar el color ni elementos de la interfaz.

Tabla 35. Mantenimiento

Característica	Valor Obtenido	%
MANTENIMIENTO	16	16
Analizabilidad	1	Excelente
Estabilidad	1	Excelente
		FIN

Con respecto a la analizabilidad, en el sistema se puede corregir funciones sin ningún problema, puede que se encuentre en la etapa del mantenimiento donde se localizan los fallos para ser corregidos
El sistema es estable y midiendo el comportamiento del administrador, usuario o sistema incluyendo el software cuando se trata de probar el software modificado o no modificado.

Tabla 36. Portabilidad

Característica	Valor Obtenido	%
PORTABILIDAD	13,2	16
Adaptabilidad	0,75	bueno
Instalación	0,5	regular
Co-existencia	1	excelente
Sustituir	1	excelente
		FIN

El sistema se puede adaptar a diferentes entornos especificados, como también el comportamiento del sistema o el usuario que está tratando de instalar el software en un entorno de usuario específico es bueno.
La instalación no es complicada, pero no ofrece la capacidad de personalizar la misma lo cual puede ser una limitación.

La coexistencia con otros programas es excelente, no presenta conflictos ni errores en este aspecto.

El comportamiento del sistema o el usuario que está tratando de usar el software con otra plataforma independiente en un entorno común compartiendo recursos comunes, es excelente porque no produce conflictos.

Tabla 37. Funcionalidad

Característica	Valor Obtenido	%
FUNCIONALIDAD	20	20
Idoneidad	1	excelente
Precisión	1	excelente
Seguridad	1	Excelente
Funcionalidad de cumplimiento	1	Excelente
		FIN

En cuanto a la idoneidad se puede afirmar que las funciones que se encuentran son las suficientes para la ejecución de las tareas, no se detectaron funciones incompletas o faltantes.

La precisión indica que los resultados obtenidos son acordes a los resultados deseados y reales.

En este factor los usuarios ven las funciones muy satisfactorias, la ocurrencia de una operación satisfactoria durante la prueba y operación de usuario del sistema, además proporcionan un resultado razonable y aceptable para lograr el objetivo específico previsto de la tarea del usuario.

La seguridad es excelente, se ve que los datos están protegidos de otros usuarios, así que, un usuario no puede ver los datos de otros usuarios con el fin de cambiarlos, borrarlos o tener acceso a datos que no deberían ser vistos por personal no autorizado.

Los resultados esperados de las diferentes tareas realizadas son razonables durante la operación del aplicativo web.

Tabla 38. Confiabilidad

Característica	Valor Obtenido	%
CONFIABILIDAD	16	16
Madurez	1	excelente
Tolerancia a fallos	1	excelente
		FIN

En cuanto a madurez, el aplicativo web no mostro fallas en ninguna de las funciones evaluadas, lo cual significa que las funciones han sido corregidas en su totalidad. Ningún fallo se detectó en la etapa de evaluación.

En cuanto a tolerancia a fallos, la capacidad del software de mantener un nivel alto de confiabilidad al no presentar fallas de operación o de vulneración de su interfaz especificada, lo convierte en un producto de muy alta calidad en esta característica.

Tabla 39. Eficiencia

Característica	Valor Obtenido	%
EFICIENCIA	11,4477	16
Tiempo de comportamiento	0,772031	Bueno
Utilización de recursos	0,677778	Bueno
		FIN

En cuanto a tiempo de comportamiento el programa es rápido para la realización de las tareas a usar con respecto a los recursos usados por el sistema.

Observaciones

Tabla 40. Observaciones

No.	Fecha	Característica	Subcaracterística	Tipo	Pregunta	Descripción
1	23/12/2010	Funcionalidad	Idoneidad	externa	Estabilidad de la especificación funcional (volatilidad)	debido a que la Auditoría se realizo cuando el software ya estaba en funcionamiento, no se tuvo conocimiento de funciones cambiadas al entrar en operación
2	23/12/2010	funcionalidad	interoperabilidad	externa	Intercambiabilidad de datos (formato de base de datos)	el sistema no tiene transferencia de datos con otros programas
3	23/12/2010	funcionalidad	interoperabilidad	externa	Intercambiabilidad de datos (intento del usuario de éxito basado en el intercambio)	El software no tiene intercambio de datos con otros programas
4	23/12/2010	funcionalidad	Seguridad	externa	Acceso auditabilidad	El software no tiene historial de acceso

No.	Fecha	Característica	Subcaracterística	Tipo	Pregunta	Descripción
5	23/12/2010	confiabilidad	Madurez	externa	Falta de densidad en contra de casos de prueba	Durante el periodo de prueba, no se obtuvo esos resultados debido a que el producto a evaluar ya está terminado
6	23/12/2010	confiabilidad	Madurez	externa	Falta de resolución	No se tiene los resultados porque se evaluó el producto final
7	23/12/2010	confiabilidad	madurez	externa	Tiempo medio entre fallos	Las pruebas al software no mostraron fallas
8	23/12/2010	usabilidad	operabilidad	externa	Accesibilidad física	Para este caso se tomó la discapacidad visual
9	23/12/2010	usabilidad	operabilidad	externa	Atractivo de interacción	Los usuarios se muestran conformes con tipo de letra, pantalla y colores
10	29/12/2010	eficiencia	Tiempo de comportamiento	externa	Rendimiento (cantidad promedio de rendimiento)	El programa solo admite una tarea
11	29/12/2010	eficiencia	Tiempo de comportamiento	externa	Rendimiento (peor ratio de rendimiento de caso)	El programa solo puede ejecutar una tarea.
12	29/12/2010	eficiencia	Tiempo de comportamiento	externa	Tiempo de vuelta (media para el tiempo de entrega)	El sistema no ejecuta tareas simultaneas
13	29/12/2010	eficiencia	Utilización de recursos	externa	Dispositivos Entrada/ Salida de utilización	El programa no utiliza muchos recursos, por lo que no provoca ineficiencias al sistema

No.	Fecha	Característica	Subcaracterística	Tipo	Pregunta	Descripción
14	29/12/2010	eficiencia	Utilización de recursos	externa	Máxima utilización de la memoria	Depende del sistema operativo
15	29/12/2010	eficiencia	Utilización de recursos	externa	Media de ocurrencia de errores de memoria	No muestra mensajes de error relacionados con la memoria
16	29/12/2010	eficiencia	Utilización de recursos	externa	Relación entre el error de memoria / hora	El sistema no muestra mensajes de error de memoria
17	29/12/2010	eficiencia	Utilización de recursos	externa	Medios de comunicación la utilización del dispositivo de equilibrio	El programa no ocupa demasiado tiempo en sincronizar recursos
18	29/12/2010	eficiencia	Utilización de recursos	externa	Utilización de la capacidad de transmisión	Solo se utilizó un usuario para transmisión de datos
19	29/12/2010	mantenimiento	analizabilidad	externa	Falta capacidad de análisis	El programa no presenta fallas de operación de usuario
20	29/12/2010	mantenimiento	analizabilidad	externa	Falta de eficiencia de análisis	El programa no presenta fallas de funcionamiento
21	29/12/2010	mantenimiento	analizabilidad	externa	Estado de capacidad de supervisión	El programa no ofrece la opción de datos de seguimiento
22	29/12/2010	mantenimiento	Modificación	externa	Cambiar la eficiencia del ciclo	El programa no tiene la capacidad de enviar solicitudes de inconformidad
23	29/12/2010	mantenimiento	Modificación	externa	Cambiar la implementación mientras transcurre el tiempo	No se pude modificar el software inmediatamente

No.	Fecha	Característica	Subcaracterística	Tipo	Pregunta	Descripción
24	29/12/2010	mantenimiento	Modificación	externa	Modificación de la complejidad	depende del programador
25	29/12/2010	portabilidad	Adaptabilidad	externa	Capacidad de adaptación de las estructuras de datos	El software no presenta cambio independiente del sistema operativo

No conformidades

Tabla 41. No conformidades

No.	Fecha	Característica	Subcaracterística	Tipo	Pregunta	No conformidad	Descripción correctiva
1	23/12/2010	usabilidad	atractivo	externa	Personalización de interfaz de aspecto	No se ofrece la opción de personalizar la apariencia	Agregar opción de modificar la apariencia con otros colores

Se observa que en la configuración personal del programa no existe el usuario, por lo tanto no puede modificar aspectos a la interface de KNOWER como son los colores. Se sugiere que se implemente una opción de modificar el entorno para el gusto del usuario, debido a que presenta colores muy claros y en días muy luminosos el contraste de colores no es el indicado.

3.4. APLICACIÓN ENCUESTA

Se decidió probar el aplicativo QUALITYSOFT con 5 personas con suficientes conocimientos en Auditoría de Sistemas y Calidad de Software (Ingeniería de Software). Todos ellos estudiantes egresados de la Universidad de Nariño del Programa de Ingeniería de Sistemas. La evaluación de los resultados se especifica a continuación:

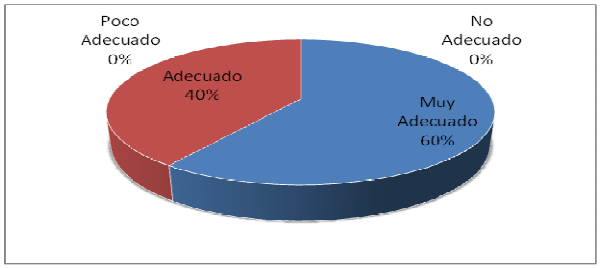
Con los resultados se hizo el siguiente análisis cuantitativo:

A la pregunta:

1. ¿Los colores que se Observan en el aplicativo QUALITYSOFT te parecen?

Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __

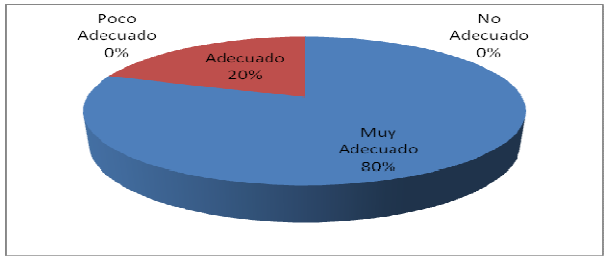
Figura 52. Resultados pregunta 1



2. ¿El tamaño y tipo de letra que se Observan en el aplicativo QUALITYSOFT te parecen?

Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __

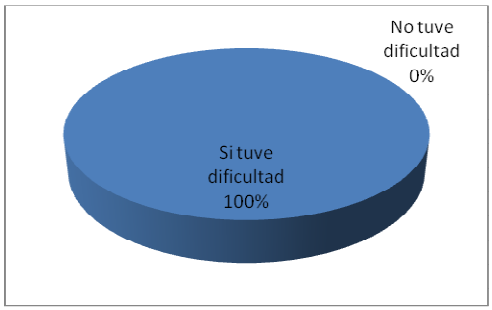
Figura 53 Resultados pregunta 2



3. ¿El aplicativo es fácil de entender o tuviste dificultad en algún momento?

Si tuve dificultad____ No tuve dificultad____

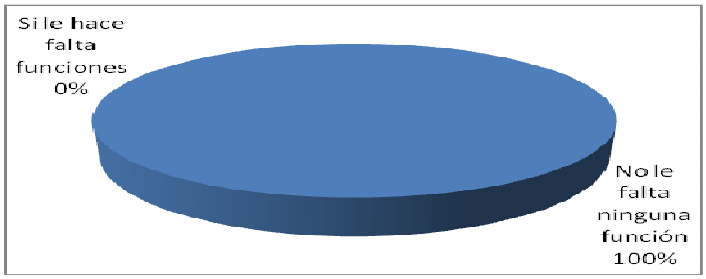
Figura 54. Resultados pregunta 3



4. ¿Las funciones del aplicativo son suficientes o crees que falta alguna función?

No le falta ninguna función ___
Si le hace falta funciones ___
Describe la función faltante _____

Figura 55. Resultados pregunta 4

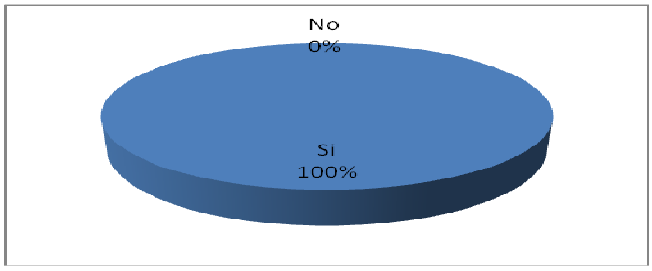


5. ¿Las métricas que se evalúan en el aplicativo QUALITYSOFT te parecen suficientes?

SI___ No___
¿En qué característica faltan métricas?

FUNCIONALIDAD __
PORTABILIDAD__
USABILIDAD__
EFICIENCIA__
MANTENIBILIDAD__
CONFIABILIDAD__
¿Cual?_____

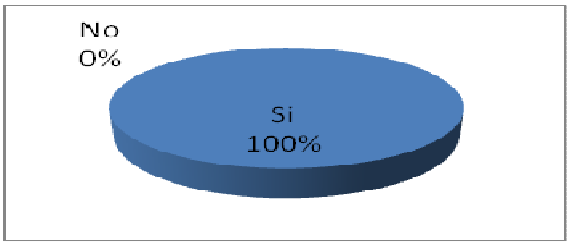
Figura 56. Resultados pregunta 5



6. ¿Consideras que hay funciones que no cumplen ningún papel?

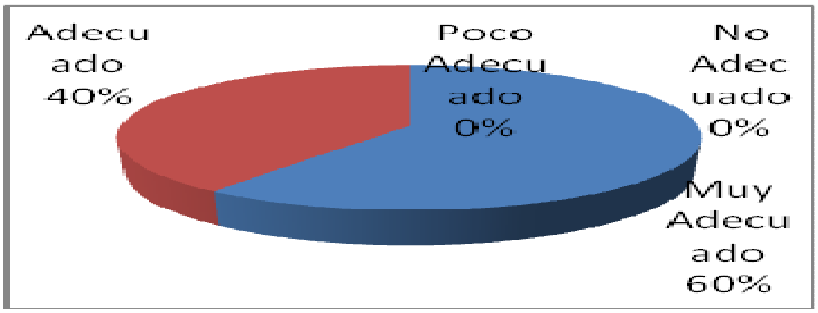
Si__ No___
¿Que
función?_____

Figura 57. Resultados pregunta 6



7. ¿El manual de usuario del aplicativo QUALITYSOFT te parece?
Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __

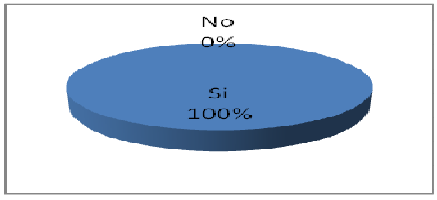
Figura 58. Resultados pregunta 7



8. ¿Te gusto el diseño del aplicativo?

Si__ No__

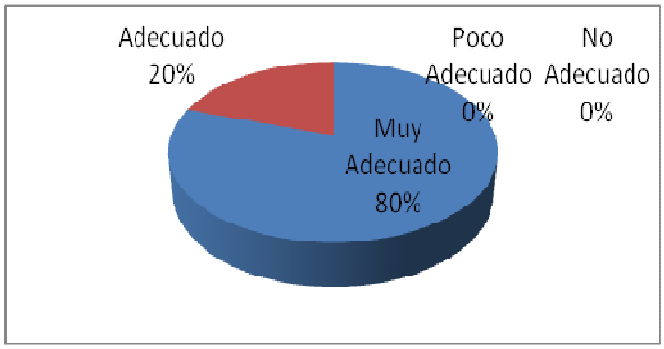
Figura 59. Resultados pregunta 8



9. ¿Los reportes que genera el aplicativo QUALITYSOFT te parecen?

Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __

Figura 60. Resultados pregunta 9



Observando los valores obtenidos en la encuesta se nota que los datos son muy favorables para el Aplicativo, se ve que el diseño, colores, letra son adecuados, las métricas a evaluar suficientes y los reportes generados son adecuados, el manual de usuario describe de manera clara y sencilla el funcionamiento del aplicativo.

4. CONCLUSIONES

La interacción con el usuario es muy intuitiva y gráfica, de fácil comprensión lo cual facilita su manejo tanto para docentes como para estudiantes.

QUALITYSOFT es desarrollado con metodología orientada a objetos para hacer posible la reutilización de código y el acoplamiento de funcionalidades que contribuyan al mejoramiento de la misma.

Los resultados obtenidos deben ser interpretados por los auditores de sistemas, con el fin de dar las explicaciones y aclaraciones de forma más claras y precisas.
Las auditorías de sistemas permiten garantizar la calidad del software luego de llevar a cabo una auditoría de calidad, es más fácil mantener un registro con las deficiencias presentadas y poder comprobar que han sido corregidas en una siguiente revisión.

Los resultados varían un poco debido a que cada auditor interpreta de manera diferente el software a auditar y tiene conceptos propios para la aprobación o desaprobación de los requisitos de evaluación del software.

La información obtenida se puede importar y exportar, lo cual facilita la movilización de los resultados obtenidos por parte de los auditores.

QUALITYSOFT ofrece la capacidad de agregar nuevas preguntas, lo cual, lo convierte en un software muy flexible al momento de la evaluación de los distintos tipos de software existentes como son el cliente-servidor, distribuidos, web.

QUALITYSOFT permite continuar con proyectos inconclusos, lo cual es una gran ventaja para los auditores, ya que el proceso de auditoría puede requerir mucho tiempo y no siempre se puede realizar una auditoría en una sola jornada de trabajo.

5. RECOMENDACIONES

Implantar el aplicativo informático QUALITYSOFT en la Universidad de Nariño para poder ser usado en la realización de auditorías al desarrollo de software y realizar la medición de calidad del software usado en la institución.

Usar el software QUALITYSOFT en el programa de Ingeniería de Sistemas en las materias de Auditoria de Sistemas y de Ingeniería de Software Aplicado el cual se convertirá en una herramienta que permita ayudar en la realización algunas de las temáticas que se desarrollan en esas materias.

Continuar con el desarrollo de QUALITYSOFT con la implementación de nuevas funcionalidades como es las pruebas de software para contar con una herramienta más completa en el proceso de Auditoria de Sistemas al desarrollo de software.

6. BIBLOGRAFIA

[1] http://definicion.de/calidad/ consultado 3 nov-2010
[2] http://es.wikipedia.org/wiki/Software consultado 3 nov-2010
[3] http://www.proyectosfindecarrera.com/que-es-una-Auditoría.htm 01 mar-2011
[4].http://148.202.148.5/cursos/cc321/fundamentos/unidad2/tema2_1.html consultado 3 nov-2010
[5] http://www.alegsa.com.ar/Dic/aplicacion.php consultado 3 nov-2010
[6] http://www.wordreference.com/definicion/inform%C3%A1tico 3 nov-2010
[7] SOLARTE SOLARTE, Francisco Nicolas Javier. MÓDULO EVALUACIÓN DE SOFTWARE. 1a ed. Universidad Nacional Abierta y a Distancia, 2010. 223 p.
[8] http://www.dccia.ua.es/jenui2004/actas/ponencias/ponencia06.pdf 17-marzo-2010
[9] Aspectos de Calidad en el Desarrollo de Software Basado en Componentes.pdf consultado 17-marzo-2010
[10]http://www.materiabiz.com/mbz/ityoperaciones/nota.vsp?nid=29672 consultado 17-marzo-2010
[11]http://www.asesoftware.com/analisis_sector_software_proexport.pdf consultado 17-marzo-2010
[12] http://www.monografias.com/trabajos/maudisist/maudisist.shtml 17-marzo-2010
[13] http://es.wikipedia.org/wiki/Organizaci%C3%B3n_Internacional_para_la_Estandarizaci%C3%B3n 17-marzo-2010
[14] INTERNATIONAL STANDARIZATION ORGANIZATION. Software engineering — Product quality. Norme international ISO/IEC 9126-1:2001. 1a ed. 2001, 33 p. 17-marzo-2010
[15] http://es.wikipedia.org/wiki/UML 17-marzo-2010
[16] http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process 17-marzo-2010
[17] http://es.wikipedia.org/wiki/C_Sharp 17-marzo-2010
[18] http://www.desarrolloweb.com/articulos/561.php 17-marzo-2010
[19] http://www.clikear.com/manuales/csharp/c10.aspx 17-marzo-2010
[20] http://es.wikipedia.org/wiki/Microsoft_Visual_Studio. 17-marzo-2010
[21] http://dev.mysql.com/doc/refman/5.0/es/features.html 17-marzo-2010
[22] http://cnx.org/content/m18938/latest/ 17-marzo-2010
[23] http://is.ls.fi.upm.es/docencia/proyecto/docs/curso/12Anexo_1_UML.doc 17-marzo-2010

[24] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/capitulo2.pdf consultado 17-marzo-2010
[25] http://es.wikipedia.org/wiki/ISO/IEC_9126 consultado 17-marzo-2010
[26]http://www.fedesoft.org/ consultado 17-marzo-2010
[27] http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad_44.pdf 17-marzo-2010
[28] PRESSMAN, Roger s."Ingeniería del Software un enfoque práctico". McGrawhill,España, 2002 22-marzo-2010
[29] Programación con visual C#.net Escrito por Francisco Charte Ojeda Publicado por Anaya Multimedia, 2002 22-marzo-2010
[30] BOOCH Grady, RUMBAUGH James, JACOBSON Ivar,"EL LENGUAJE UNIFICADO DE MODELADO", Madrid 1999. 22-marzo-2010

ANEXO A. MANUAL DE INSTALACIÓN

El objetivo de este manual es explicar la instalación. Se dan una serie de nociones en la instalación y configuración de todas las aplicaciones necesarias para el correcto funcionamiento del aplicativo.

Se debe comentar que el sistema es para la plataforma Windows.

A continuación se comenta los programas necesarios para que funcione el sistema QUALITYSOFT.

Programas Necesarios

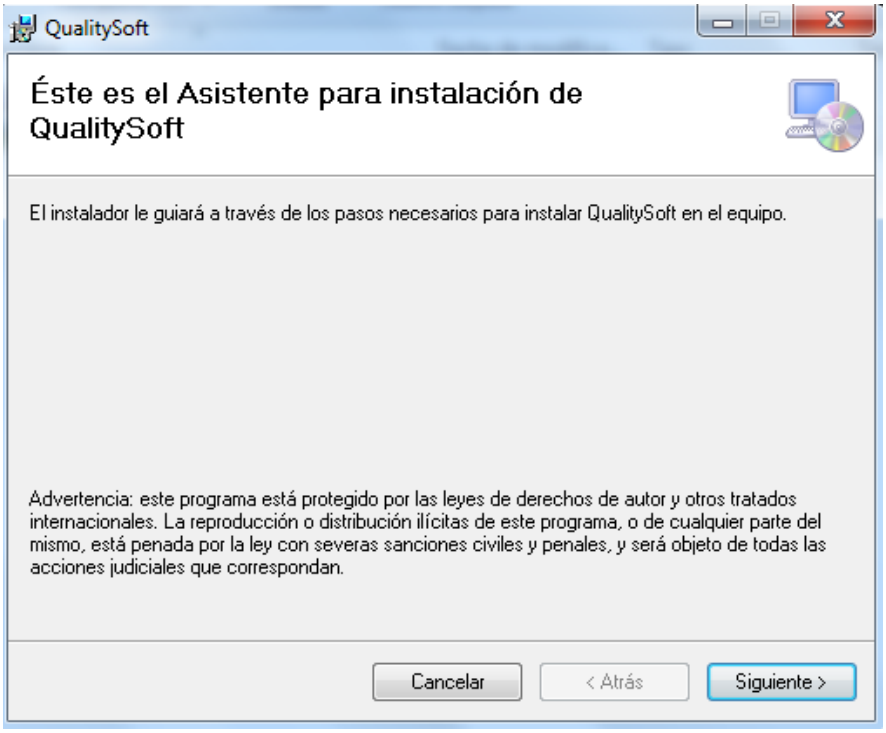
Primeramente, se muestran los enlaces donde se pueden encontrar tanto los paquetes como toda la información para la correcta instalación de todas las aplicaciones necesarias para el correcto funcionamiento de nuestro aplicativo.

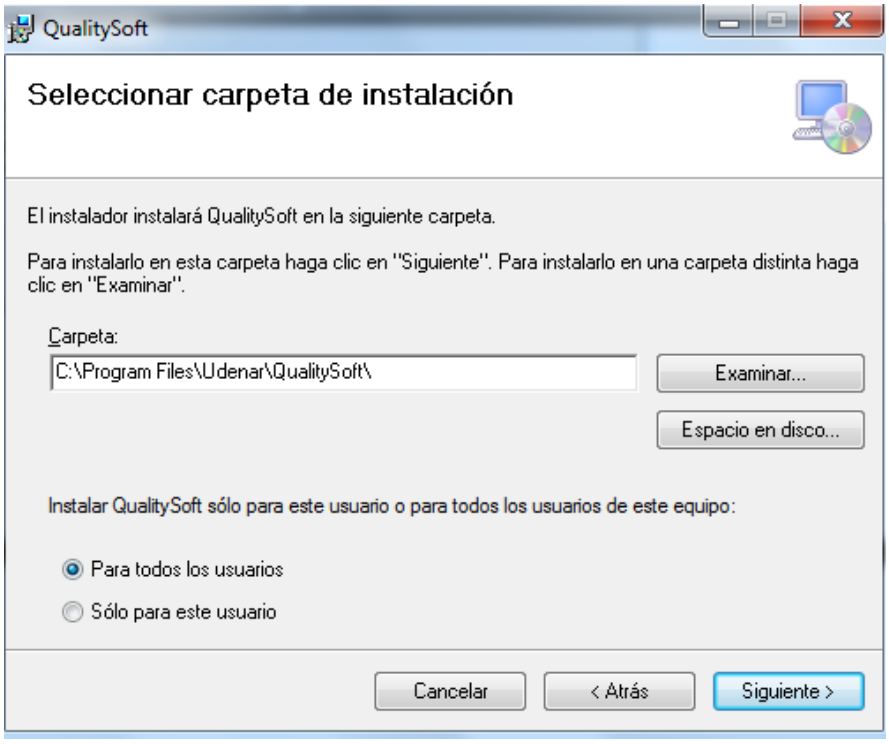
- Framework 4.0 http://www.microsoft.com/downloads/en/
- MySQL 5.0 www.mysql.com

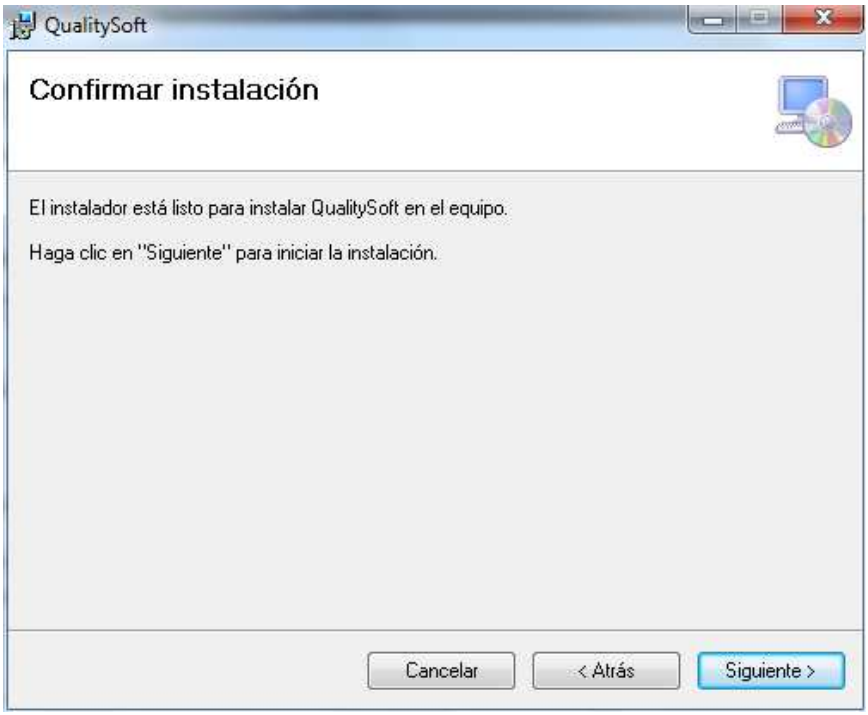
Se recomienda leer los manuales de las distintas aplicaciones para obtener más detalles sobre la instalación de éstas.

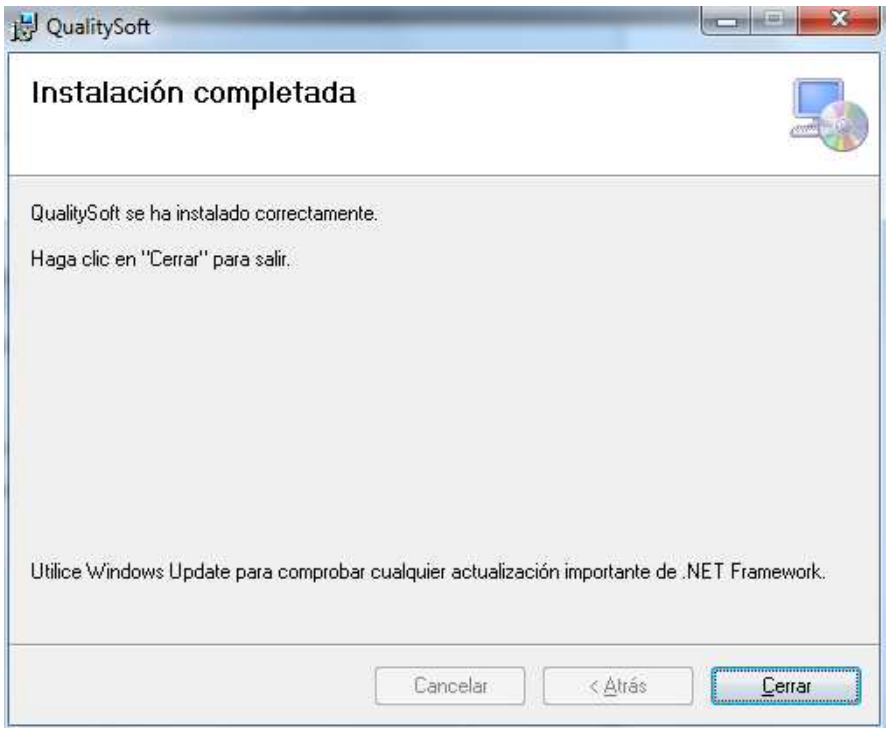
Instalación del aplicativo QUALITYSOFT

Procedemos a instalar el aplicativo y para ello, seguimos los siguientes pasos:


QualitySoft
Éste es el Asistente para instalación de QualitySoft
El instalador le guiará a través de los pasos necesarios para instalar QualitySoft en el equipo.
Advertencia: este programa está protegido por las leyes de derechos de autor y otros tratados internacionales. La reproducción o distribución ilícitas de este programa, o de cualquier parte del mismo, está penada por la ley con severas sanciones civiles y penales, y será objeto de todas las acciones judiciales que correspondan.
Cancelar
< Atrás
Siguiente >


QualitySoft
Seleccionar carpeta de instalación
El instalador instalará QualitySoft en la siguiente carpeta.
Para instalarlo en esta carpeta haga clic en "Siguiente". Para instalarlo en una carpeta distinta haga clic en "Examinar".
Carpeta:
C:\Program Files\Udenar\QualitySoft\
Examinar...
Espacio en disco...
Instalar QualitySoft sólo para este usuario o para todos los usuarios de este equipo:
Para todos los usuarios
Sólo para este usuario
Cancelar
< Atrás
Siguiente >


QualitySoft
Confirmar instalación
El instalador está listo para instalar QualitySoft en el equipo.
Haga clic en "Siguiente" para iniciar la instalación.
Cancelar
< Atrás
Siguiente >


QualitySoft
Instalación completada
QualitySoft se ha instalado correctamente.
Haga clic en "Cerrar" para salir.
Utilice Windows Update para comprobar cualquier actualización importante de .NET Framework.
Cancelar
< Atrás
Cerrar

Una vez terminada la instalación, se procede a crear la Base de Datos en MySQL 5.0, seguimos los siguientes pasos:

- Creamos la base de datos llamada: auditoria
- Creamos las tablas, ejecutamos el script.sql

Con estos pasos tenemos configurada la base de datos.

Por defecto trae un usuario para entrar al programa QUALITYSOFT como administrador:
IDENTIFICADOR: admin
CLAVE: 123

ANEXO B. MANUAL DE USUARIO

QUALITYSOFT es una herramienta de auditoría de sistemas que permite evaluar la calidad de desarrollo de software basado en la norma de calidad ISO/IEC 9126.
Con QUALITYSOFT podrá crear nuevo proyecto, además permite ingresar datos del auditor, seleccionar factores (indicadores) y evaluar las métricas.
También Abrir proyecto permite visualizar todos los proyectos ya realizados, seguir desarrollando proyectos incompletos, también crear copias de seguridad de los proyectos y restaurarlos y mostrar un informe sobre estos, y borrar proyectos.

El objetivo que se persigue con la aplicación del presente manual es dar a conocer a los usuarios finales las características y las formas de funcionamiento de la herramienta.

REQUERIMIENTOS BÁSICOS HARDWARE

Computadora con memoria RAM 512 mega bytes.

REQUERIMIENTOS BÁSICOS SOFTWARE

Tener instalado MySQL 5.0
Tener instalado Framework 4.0
Plataforma Windows

APLICACIÓN

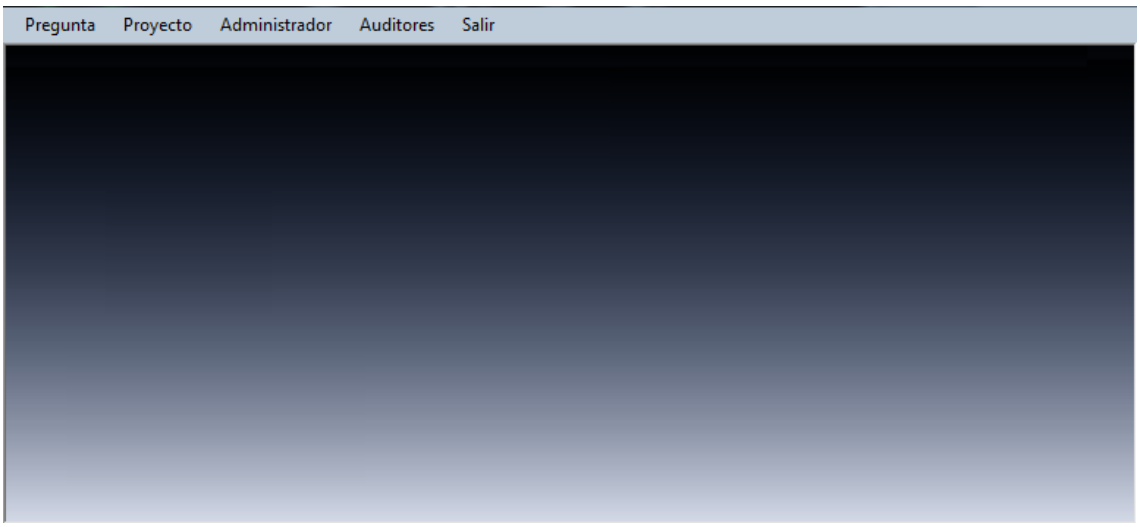
INICIO – LOGIN

Ejecutamos la aplicación y nos aparece una ventana de identificación de usuarios, aquí seleccionamos el tipo de usuario si es Auditor o Administrador, después se digita el nombre y clave.

Si la clave o nombre son incorrectos aparece un mensaje de error.

COMO ADMINISTRADOR

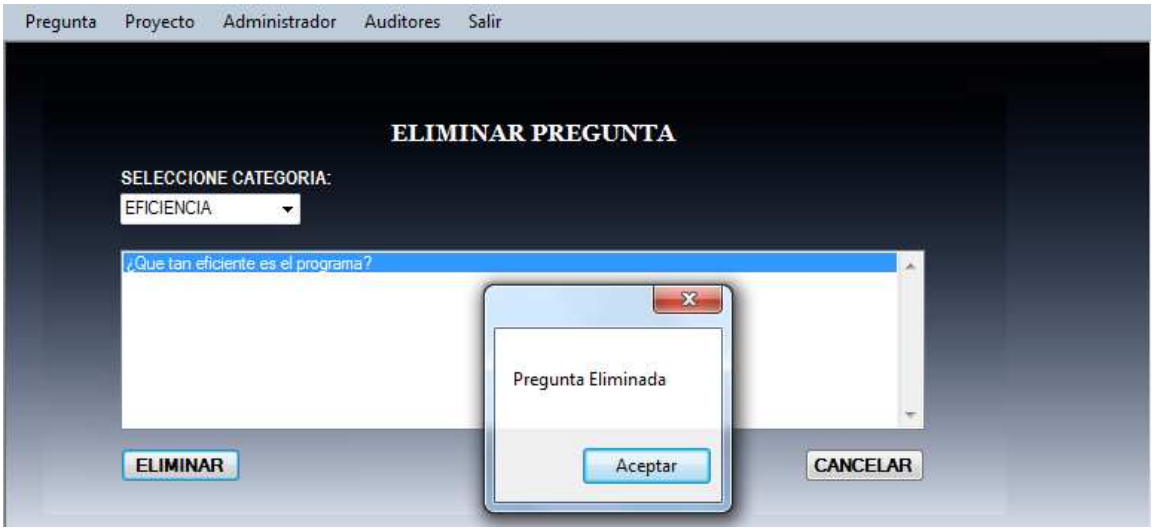
Cuando entramos al sistema como administrador muestra un menú con las opciones de Pregunta, Proyecto, Administrador, Auditores y Salir.



Menú - Pregunta:
se tiene 2 opciones AGREGAR y ELIMINAR preguntas.

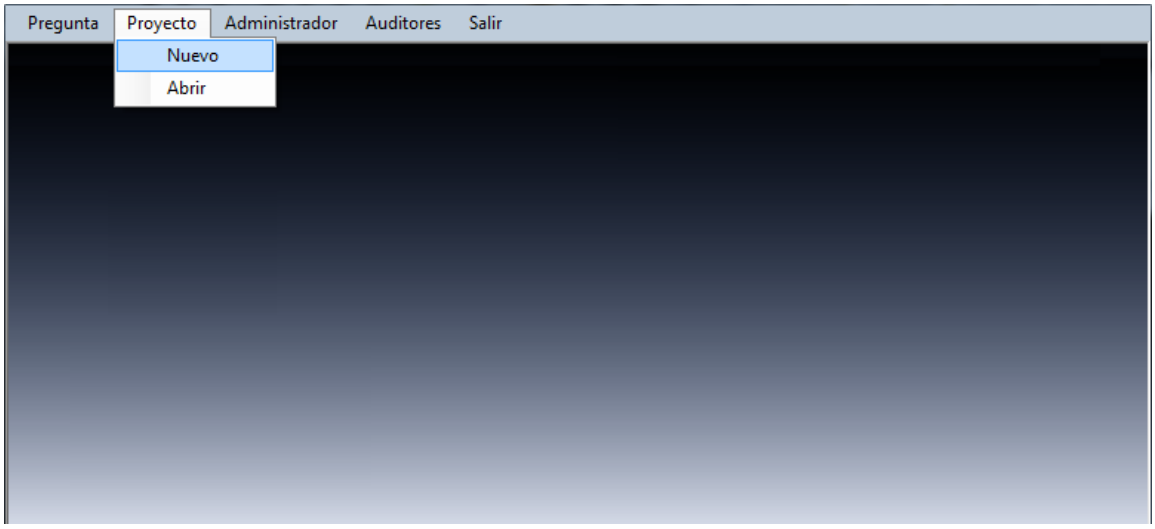
Agregar: selecciona alguno de los 6 factores, y después se escribe la pregunta, finalmente clic en botón AGREGAR para guardar los datos.

Eliminar: Seleccionamos el factor y muestra las preguntas referentes a ese factor, se selecciona la pregunta a eliminar y se da clic en botón ELIMINAR para confirmar la tarea.



Menú - Proyecto:

Se encuentra 2 opciones: Nuevo, Abrir.



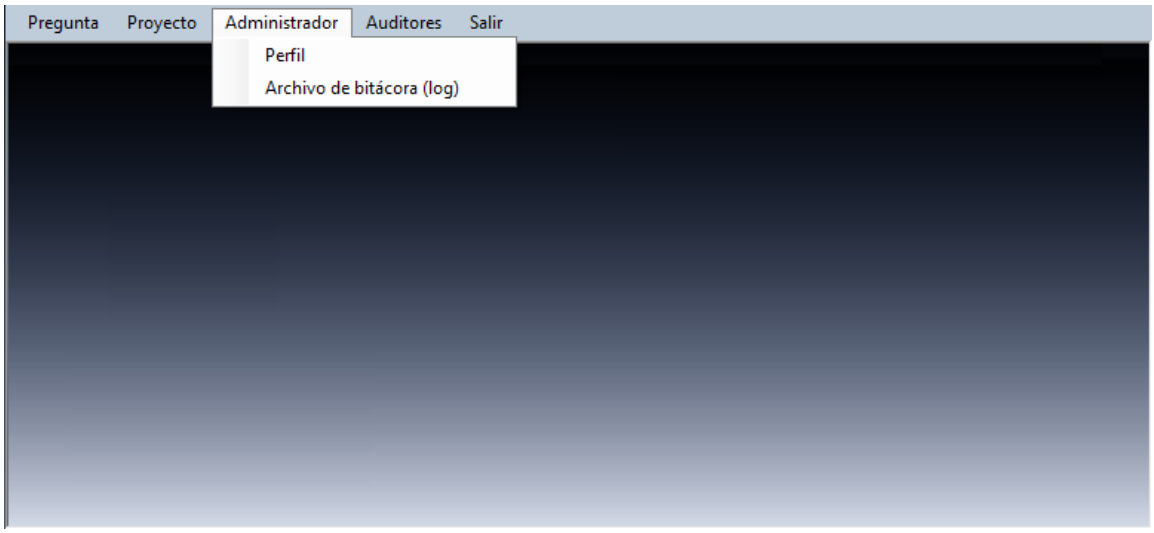
Nuevo: ir a NUEVO PROYECTO.

Abrir: Encontrara todos los proyectos de los auditores, indicando nombres, fecha y hora cuando fue creado, también se encontrara con un menú en la parte inferior: Mostrar, Continuar, Backup BD, Restaurar BD, Cerrar proyecto.

Primero se selecciona algún proyecto,

Mostrar: Muestra un reporte detallado del proyecto.
Continuar: Puede seguir desarrollando un proyecto incompleto.
Backup BD: Hace un copia de los proyectos, aquí aparece una ventana para guardar su copia de seguridad de la base de datos.
Restaurar BD: Restaura la base de datos, aparece una ventana para seleccionar la copia que desee restaurar.
Cerrar proyecto: Borra el proyecto de la lista.

Menú - Administrador:



Perfil: muestra los datos personales del administrador los cuales y puede modificarlos, para confirmar se da clic en botón Aceptar.

Si existe algún error al registrar algún dato, el sistema te avisa con un mensaje de error.

Archivo de bitácora (log): muestra un registro de la actividad realizada en el aplicativo.

Menú - Auditores: aquí se encuentra los perfiles de todos los auditores, además se puede ingresar un nuevo auditor (casillas blancas), se puede modificar algún campo y para eliminar un auditor se selecciona toda la fila y presionamos botón Supr (suprimir) y se le da clic en botón Aceptar para confirmar la acción.

Si existe algún error al introducir algún dato, el sistema te avisa con un mensaje de error.

COMO AUDITOR

LOGIN

Si la clave o nombre son incorrectos aparece un mensaje de error.

Cuando se entra al sistema como Auditor muestra un menú con las opciones de NUEVO PROYECTO y ABRIR PROYECTO

Nuevo Proyecto: ir a Nuevo Proyecto.

Abrir Proyecto: se encuentra los proyectos creados por el auditor, se observa el nombre del proyecto, fecha y hora, además hay un menú en la parte inferior con la opción de Mostrar y Continuar.

Primero se selecciona el proyecto,

Mostrar: muestra un reporte del proyecto.
Continuar: puedes seguir desarrollando proyectos incompletos.

NUEVO PROYECTO:

Aquí se ingresa el nombre del proyecto.
También se ingresa el nombre de la empresa (Campo Opcional)

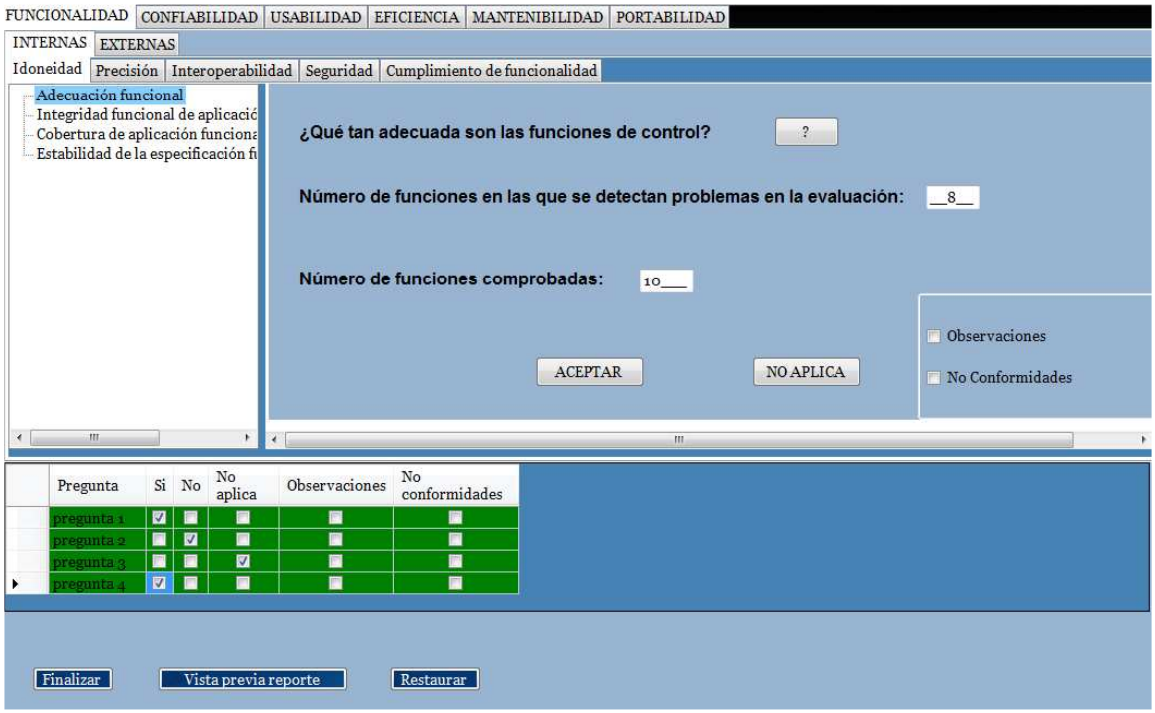
SELECCIÓN DE FACTORES:

Una vez que se ingresa el nombre de auditoría, damos clic en botón Aceptar, esto lleva a una ventana donde se selecciona los factores a evaluar, depende del criterio de cada evaluador.

Una vez seleccionados los factores se ingresa su porcentaje (%) de interés (criterio del evaluador), el porcentaje total debe ser 100%.

Si el porcentaje es incorrecto el programa te mostrará un mensaje de porcentaje erróneo.

Cuando ya hemos seleccionado los factores con su respectivo porcentaje de interés, damos clic en siguiente y nos aparece un menú con los factores seleccionados y sus respectivas métricas internas y externas.



Se procede a realizar la evaluación del proyecto, se puede observar los factores en el menú superior, después el tipo de métrica interna o externa en el menú intermedio y las subcaracterísticas.

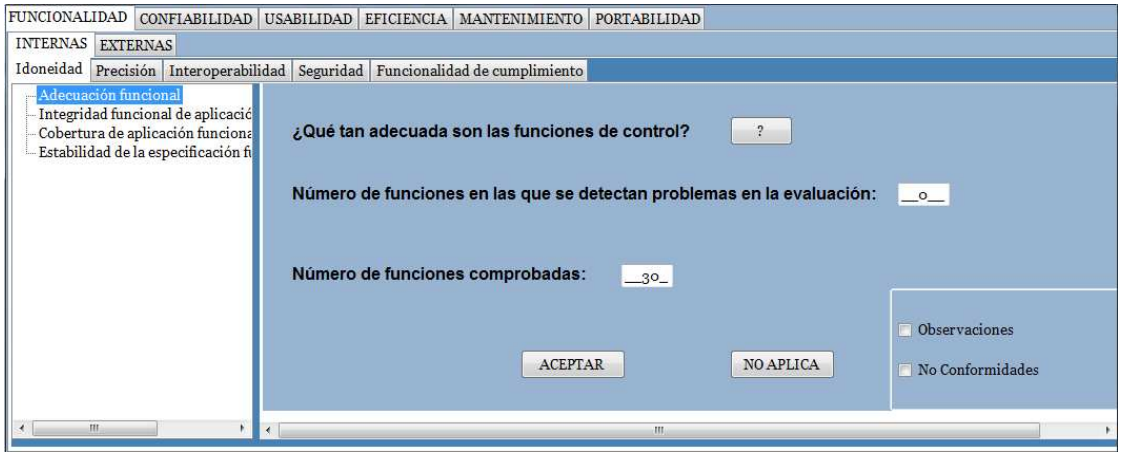
En la parte inferior se enuentra las preguntas cualitativas y unos botones FINALIZAR y RESTAURAR.

Botón Finalizar: guarda todos los datos referentes al factor que se está desarrollando, cuando das clic en finalizar no se puede volver a realizar este factor.

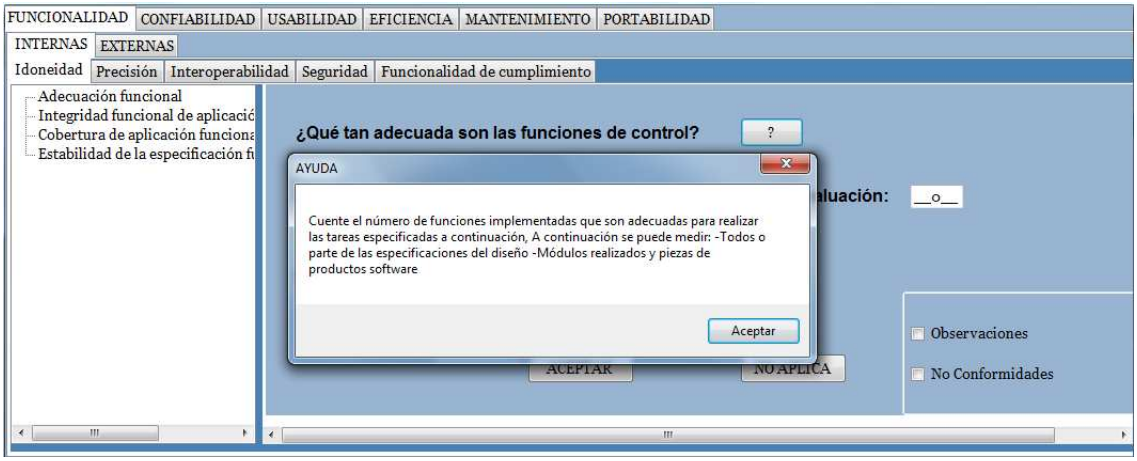
Botón Restaurar: es para las preguntas cualitativas, restará la lista al estado inicial.

Botón Vista previa reporte: muestra un reporte parcial del proyecto.

Cuando seleccionas una subcaracterística, esta despliega una serie de preguntas así:



Cuando se da clic en alguna pregunta, te aparece una ventana con sus atributos y entre ellos unos botones como son: ACEPTAR, NO APLICA, AYUDA (?) y lista de chequeo, que se puede realizar.



Cuando presionas Botón AYUDA (?), te dice como la debes hacer, es una guía que indica el procedimiento para levantar la información requerida en la pregunta.

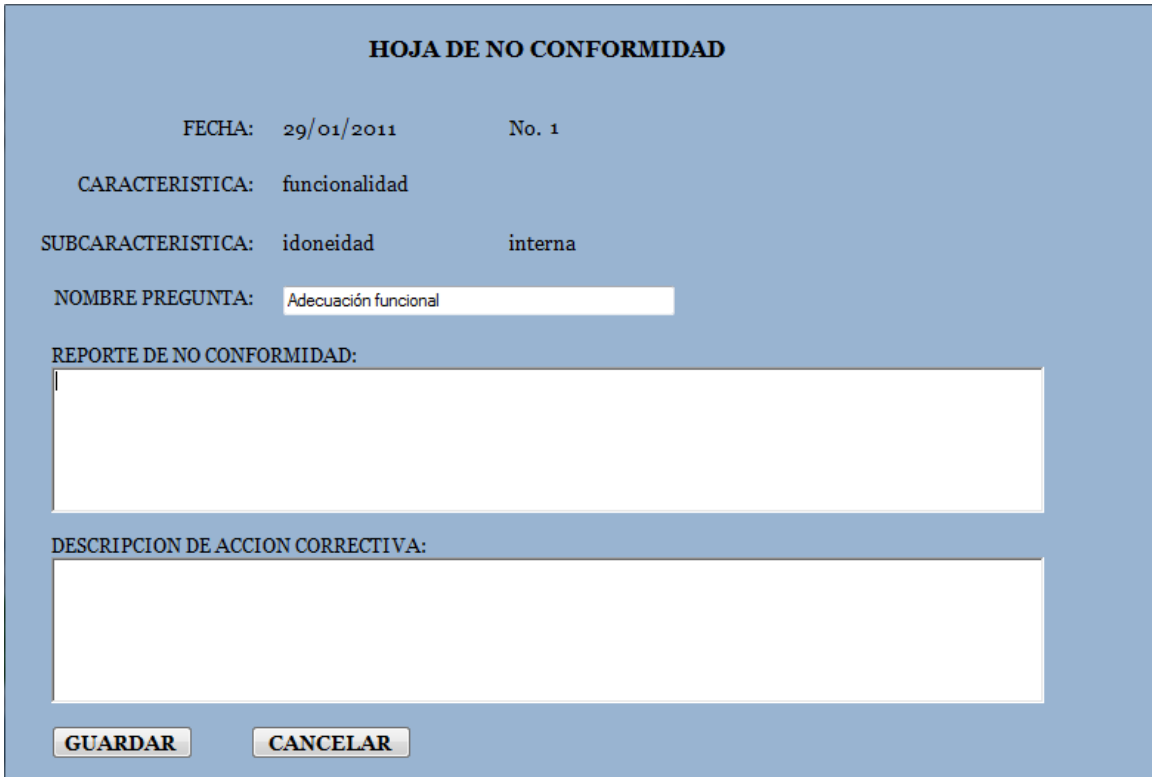
Cuando presionas Botón Aceptar, has evaluado realizado la métrica, se guardan los datos y no podrás volver a modificar.

Cuando presionas Botón No Aplica, la pregunta no se adapta a tus necesidades y no la puedes volver a modificar.

Si se da clic en la lista de chequeo, se abre una ventana en la cual vas a llenar un informe de NO CONFORMIDAD u OBSERVACIONES acerca de la pregunta que estas evaluando.

OBSERVACIONES, aquí se ingresa los datos como son: fecha, característica, subcaracterísticas, tipo, nombre pregunta e información correspondiente para cada caso.

Una vez listo el informe se da clic en GUARDAR para confirmar la acción y se cierra automáticamente la ventana o das clic en CANCELAR, el cual te cierra la ventana sin guardar datos.



NO CONFORMIDAD, se ingresan los datos como son: fecha, característica, subcaracterísticas, tipo, nombre pregunta e información correspondiente para cada caso.

Una vez listo el informe das clic en GUARDAR para confirmar acción y se cierra automáticamente la ventana o das clic en CANCELAR, el cual te cierra la ventana sin guardar datos.

ANEXO C. LISTA DE METRICAS

METRICAS EXTERNAS

¿Qué tan completa es la aplicación conforme a las especificaciones de requisitos?
¿Qué tan correcta es la aplicación funcional?
¿Qué tan estable es la especificación funcional después de entrar en operación?
¿Las diferencias entre los resultados reales y razonables esperados son aceptables?
¿Con qué frecuencia los usuarios finales encuentran resultados inexactos?
¿Con qué frecuencia los usuarios finales encuentran resultados con precisión insuficiente?
¿Se han puesto en marcha correctamente las funciones de interfaz a cambio de la transferencia de datos especificado?
¿Con qué frecuencia el usuario final no logró intercambio de datos entre el objetivo software y otros programas?
¿Qué tan completa es la pista de auditoría sobre el acceso del usuario al sistema y los datos?
¿Qué tan controlable es el acceso al sistema?
¿Cuál es la frecuencia de los fenómenos de corrupción de datos?
¿Es compatible con la funcionalidad del producto que venga con los reglamentos, normas y convenciones?
¿Son compatibles las interfaces conforme a las normas aplicables, las normas y convenios?
¿Cuántos problemas todavía existen, que puedan surgir como fallas en el futuro?
¿Cuántos fallos fueron detectados durante el período de prueba definido?
¿Cuántas condiciones de fracaso se resuelven?
¿Cuántos defectos fueron detectados durante el período de prueba definido?
¿Cuántos defectos se han corregido?
¿Con qué frecuencia el software falla en la operación?
¿Qué cantidad de casos de prueba requeridos han sido ejecutados durante la prueba?
¿Está el producto bien probado?
¿Con qué frecuencia el producto de software causa el desglose del producto total?

¿Cuántos patrones de fallo, fueron puestos bajo control para evitar fracasos críticos y graves?
¿Cuántas funciones se implementan con la capacidad de evitar operaciones incorrectas?
¿Cómo está el sistema disponible para su uso durante el período de tiempo determinado?
¿Cuál es el tiempo promedio en que el sistema se mantiene disponible cuando se produce un error antes del inicio gradual?
¿Cuál es el tiempo promedio necesario del sistema, para completar la recuperación del sistema?
¿Con qué lapso de tiempo el sistema puede reiniciar la prestación del servicio a los usuarios dentro de un tiempo determinado?
¿Qué tan capaz es el producto en la restauración de evento en sí mismo o a petición después de anormalidades o fallas?
¿Qué tan efectiva es la capacidad de restauración?
¿Cómo cumple con la fiabilidad del producto que proceda con reglamentos, normas y convenciones?
¿Qué cantidad de funciones (o tipos de funciones) se entiende después de leer la descripción del producto?
¿Qué cantidad de tutoriales del producto software pueden acceder los usuarios?
¿Qué cantidad de funciones, el usuario puede operar con éxito después de una demostración o instrucción?
¿Qué cantidad de funciones (o tipos de función) puede identificar el usuario, basada en condiciones de puesta en marcha?
¿Qué cantidad de funciones del producto, el usuario es capaz de entender correctamente?
¿Cuánto tiempo le toma al usuario aprender a usar una función?
¿Cuánto tiempo le toma al usuario aprender como realizar la tarea especificada eficientemente?
¿Qué cantidad de tareas se puede completar correctamente después de utilizar la documentación?
¿Qué cantidad de temas de ayuda puede localizar al usuario?
¿Con qué frecuencia un usuario tiene que acceder a la ayuda para aprender el funcionamiento?
¿Cuán consistente es el componente de la interfaz de usuario?
¿Puede el usuario corregir fácilmente los errores en las tareas?
¿ Puede el usuario recuperar su entrada fácilmente?
¿Puede el usuario seleccionar fácilmente los valores de parámetros para una operación pertinente?

¿Puede el usuario comprender fácilmente los mensajes del sistema de software?
¿En qué cantidad, las condiciones de error significan que el usuario proponga la acción correcta de recuperación?
¿Puede el usuario recuperar fácilmente errores en el software antes que empeoren?
¿Puede el usuario operar el software el tiempo suficiente sin cometer errores?
¿Con qué frecuencia el usuario corrige con éxito los errores de entrada?
¿Puede el usuario personalizar fácilmente los procedimientos de operación para su conveniencia?
¿Puede fácilmente reducir los procedimientos de operación de usuario para su conveniencia?
¿Qué cantidad de funciones pueden ser accedidas por los usuarios con discapacidad física?
¿Qué tan atractiva es la interfaz para el usuario?
¿Qué cantidad de elementos de la apariencia de la interfaz se pueden personalizar a la satisfacción del usuario?
¿Cómo hace el software para estar acorde con las normas, convenciones, guías de estilo o regulaciones?
¿Cuál es el tiempo necesario para completar una determinada tarea?
¿Cuál es el límite promedio de tiempo que se requiere en el cumplimiento de una función?
¿Cuántas tareas pueden realizarse con éxito en un período determinado de tiempo?
¿Cuál es el límite promedio del sistema en términos del número y manejo de tareas simultáneas?
¿Cuál es el tiempo de espera que el usuario requiere después de ejecutar una instrucción?
¿Cuál es el tiempo promedio de espera según las experiencias de usuario después de ejecutar una instrucción?
¿Cuál es el límite absoluto de tiempo que se requiere en el cumplimiento de una tarea de trabajo?
¿Qué cantidad de tiempo esperan los usuarios hasta la respuesta del sistema?
¿La utilización de dispositivos Entrada y Salida es demasiado alta, provocando ineficiencias?
¿Cuál es el límite absoluto de utilización dispositivos de Entrada / Salida en el cumplimiento?
¿Cuál es el impacto en la utilización de dispositivos de Entrada / Salida, tiempos de espera del usuario?

¿Cuál es el límite absoluto de memoria necesaria en el cumplimiento de una función?
¿Cuántos errores de memoria sobrepasaron el tiempo máximo establecido?
¿Cuál es el límite absoluto de las transmisiones necesarias para cumplir una función?
¿Cuál es el número medio de mensajes de error relacionadas con la transmisión y fallos durante un período de tiempo especificado y la utilización de determinada?
¿El sistema de software es capaz de realizar tareas dentro de lo esperado?
¿Cómo cumple la eficiencia del producto de acuerdo a regulaciones, normas y convenciones?
¿Puede el usuario identificar una operación específica que causó la anomalía?
¿Qué capacidad tienen las funciones de diagnóstico en el apoyo a un análisis causal?
¿ Se puede encontrar fácilmente por mantenimiento a la causa de la falla?
¿Puede el usuario analizar eficientemente la causa de la anomalía?
¿Puede el problema del software ser resuelto a satisfacción dentro de un plazo aceptable?
¿Puede el administrador cambiar fácilmente el software para resolver el problema de la anomalía?
¿Puede el administrador cambiar fácilmente el software para resolver un problema de software?
¿Puede el usuario o el desarrollador cambiar fácilmente los parámetros de los ajustes de software?
¿El usuario puede identificar fácilmente versiones revisadas?
¿Puede el software de usuario operar el sistema sin fallas después de un arreglo?
¿Cómo se cumple con el mantenimiento del producto que proceda regulaciones, normas y convenciones?
¿El usuario puede operar el sistema de software sin errores después del mantenimiento?
¿Puede el usuario o desarrollador de software adaptarse fácilmente al medio ambiente informático ?
¿El usuario puede fácilmente instalar o administrar el software para operaciones de medio ambiente informático?
¿Puede el usuario o administrador volver fácilmente a intentar la instalación de configuración de software?
¿Con qué frecuencia el software sufre algunas limitaciones o fallas inesperadas al operar simultáneamente?
¿Puede el usuario o administrador seguir utilizando los mismos datos después de sustitución de esa versión del software?

¿La migración del sistema de software de una versión a otra pasa con éxito?
¿Existente coherencia entre la interfaz de usuario con los nuevos componentes?
¿Cómo se cumple la portabilidad del producto con la normativa aplicable, normas y convenciones?
¿Puede el usuario o desarrollador de software adaptarse fácilmente al manejo de datos en el nuevo entorno?

METRICAS INTERNAS

¿Qué tan adecuada son las funciones de control?
¿Qué tan completa es la aplicación funcional?
¿Qué tan correcta es la aplicación funcional?
¿Qué tan estable es la especificación de funciones durante el desarrollo del ciclo de vida?
¿Cómo han sido los requisitos de precisión en la práctica?
¿Qué tan completa es la aplicación de niveles específicos de precisión de los datos?
¿Se ha puesto correctamente en marcha los formatos de datos de interfaz?
¿Se ha puesto correctamente en marcha los protocolos de interfaz?
¿Son auditable las entradas de acceso?
¿Qué tan controlable es el acceso al sistema?
¿Qué tan completa es la aplicación de Prevención de la corrupción de datos?
¿Qué tan completa es la aplicación de cifrado de datos?
¿Con las normas aplicables, es compatible la funcionalidad del producto con las normas y las convenciones?
¿Con las normas aplicables, cómo son las interfaces conforme a las normas y convenios?
¿Cuántos defectos fueron detectados en el producto revisado?
¿Cuántos defectos se han corregido en el producto revisado?
De los casos de prueba necesarios, ¿Qué porcentaje están cubiertos por el plan de pruebas?
¿Cuántos patrones de fallo, fueron puestos bajo control para evitar fracasos críticos y graves?
¿Cuántas funciones se realizan, para evitar capacidad con operaciones incorrectas?
¿Qué tan capaz es el producto de auto restaurarse después de fallas?
¿Qué tan efectiva es la capacidad de auto restauración?
¿Qué tan compatible es la confiabilidad del producto con los reglamentos aplicables, las normas y las convenciones?
¿Qué cantidad de funciones (o tipos de función) se describen en el producto?
¿Qué cantidad de funciones requieren demostración de capacidad?
¿Qué cantidad de funciones del producto son evidentes para el usuario?
¿Qué cantidad de funciones del producto, el usuario es capaz de entender correctamente?

¿Qué cantidad de funciones están descritas en la documentación del usuario y / o ayudas de las instalaciones?
¿Qué cantidad de elementos de entrada son verificados para proporcionar datos válidos?
¿Qué cantidad de funciones pueden ser canceladas antes de terminar una operación?
¿Qué cantidad de funciones se puede deshacer?
¿Qué cantidad de funciones se pueden personalizar en operación?
¿Qué cantidad de funciones se pueden personalizar, para el acceso por los usuarios con discapacidades físicas?
¿Qué cantidad de funciones tienen la condición de las operaciones de capacidad de seguimiento?
¿Qué cantidad de operaciones se comportan de la misma manera a otras operaciones en diferentes partes del sistema?
¿Qué cantidad de mensajes se explican por sí mismo?
¿Qué cantidad de elementos de la interfaz se explican por sí mismo?
¿Qué cantidad de funciones pueden tolerar un error del usuario?
¿Es atractiva la interfaz para el usuario?
¿Qué cantidad de elementos de la interfaz de usuario se pueden personalizar en apariencia?
¿El producto cumple con los reglamentos aplicables, las normas y convenios para la usabilidad?
¿Cuál es el tiempo estimado para completar una determinada tarea?
¿Cuál es el número estimado de las tareas que se pueden realizar en más de una unidad de tiempo?
¿Cuál es el tiempo estimado para completar un grupo de tareas relacionadas?
¿Cuál es la estimación de uso para completar una determinada tarea?
¿Cuál es el tamaño de la memoria que el producto va a ocupar para completar una determinada tarea?
¿Cuál es la densidad de los mensajes relacionados con la utilización de memoria en las líneas de código?
¿Cuál es el importe estimado de transmisión de la utilización de los recursos?
¿Cómo cumple la eficiencia del producto que proceda reglamentos, normas y convenciones?
¿Cómo es la grabación completa del estado del sistema?
¿Cómo es la prestación de las funciones de diagnóstico?
¿Los cambios en las especificaciones y los módulos del programa son registrados adecuadamente en el código?
¿Cuál es la frecuencia de resultados adversos después de la modificación?

¿Qué tan grande es el impacto de la modificación en el producto de software?
¿Qué tan completa es la capacidad de prueba incorporada?
¿Puede el software hacer la prueba de forma independiente?
¿Cómo se completa el resultado de la prueba durante la prueba muestra?
¿Se cumple el mantenimiento del producto que proceda reglamentos, normas y convenciones?
¿Es el producto adaptable a los cambios de estructura de datos?
¿Cómo se adapta el producto al cambio de Hardware / Redes relacionadas con el medio ambiente?
¿Cómo se adapta el producto a un cambio organizacional?
¿Cómo es el esfuerzo para llevar a cabo operaciones en la portabilidad producto?
¿El producto es adaptable a software del sistema relacionado con los cambios ambientales?
¿Es fácil repetir la operación de instalación?
¿Qué nivel de esfuerzo se requiere para la instalación?
¿Qué tan flexible y adaptable es la capacidad de la instalación?
¿Qué tan flexible es el producto de compartir su entorno con otros productos sin resultados adversos?
¿Cuál es la cantidad de datos originales que se mantienen sin cambios después de la reposición con este producto?
¿Cuál es la cantidad de funciones que no cambian?
¿Cómo cumple la portabilidad del producto a las normas aplicables, las normas y las convenciones?

ANEXO D. ENCUESTA USUARIO

1. **¿Los colores que se Observan en el aplicativo QUALITYSOFT te parecen?**

Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __

2. **¿El tamaño y tipo de letra que se Observan en el aplicativo QUALITYSOFT te parecen?**

Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __

3. **¿El aplicativo es fácil de entender o tuviste dificultad en algún momento?**

Si tuve dificultad____ No tuve dificultad____

4. **¿Las funciones del aplicativo son suficientes o crees que falta alguna función?**

No le falta ninguna función ___
Si le hace falta funciones ___
Describe la función faltante
_____
_____
_____

5. **¿Las métricas que se evalúan en el aplicativo QUALITYSOFT te parecen suficientes?**

SI___ No___
¿En qué característica faltan métricas?
FUNCIONALIDAD __
PORTABILIDAD__
USABILIDAD__
EFICIENCIA__
MANTENIBILIDAD__
CONFIABILIDAD__
¿Cual?_____

6. **¿Consideras que hay funciones que no cumplen ningún papel?**

Si__ No___
¿Que función?_____
_____

7. **¿El manual de usuario del aplicativo QUALITYSOFT te parece?**

Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __

8. **¿Te gusto el diseño del aplicativo?**

Si__ No__

9. **¿Los reportes que genera el aplicativo QUALITYSOFT te parecen?**

Muy Adecuados __ Adecuados __ Poco Adecuados __ No Adecuados __