Universidad de las Ciencias Informáticas

Facultad 3

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

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Daylín Dainery Montero Ramos

Juliet García Tamayo

Tutor(es):MSc. Asnier Enrique Góngora Rodríguez

Ing. Lisandra Díaz Figueredo

La Habana, Cuba

Curso 2012-2013

El hombre puede hacer de si mismo muchas cosas producto de su propio esfuerzo físico y espiritual, el que se proponga cultivar la virtud la cultiva, el que se proponga alcanzar los más altos niveles de conocimiento los alcanza.

Fidel Castro Ruz

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de _____del año ____.

_____	_____
Daylín Dainery Montero Ramos	Juliet García Tamayo
(Autor)	(Autor)
_____	_____
MSc. Asnier Enrique Góngora Rodríguez	Ing. Lisandra Díaz Figueredo
(Tutor)	(Cotuto

Datos de Contacto

Tutor: MSc. Asnier Enrique Góngora Rodríguez.

Categoría docente: Asistente

Síntesis: Graduado en la Universidad de Ciencias Informáticas en el año 2009. Años de experiencia en el tema de calidad: 6.

Categoría Científica: Máster

Correo electrónico: agongora@uci.cu.

Teléfono: 8358877.

Cotutor: Ing. Lisandra Díaz Figueredo.

Síntesis: Graduado en la Universidad de las Ciencias Informáticas, Especialista General.

Correo electrónico: lfigueredo@uci.cu.

Autor(es): Daylín Dainery Montero Ramos

Correo electrónico: ddmontero@estudiantes.uci.cu

Teléfono: 8372283

Autor(es): Juliet García Tamayo

Correo electrónico: jgarciat@estudiantes.uci.cu

Teléfono: 8372283

Agradecimientos

De Daylín:

A mi mamá y mi papá, por darme la vida, por permitirme ser quien soy y aunque están lejos cumpliendo con la Revolución contribuyeron y me inspiraron a hacer realidad mis sueños.

A mi hermano y mis hermanas por apoyarme siempre y brindarme su cariño, y por toda su ayuda en la ausencia de mis padres.

A mi abuela Margo a quien adoro, por complacerme en todos mis gustos.

A mis tías y tíos por darme ánimo y apoyo en todo momento,

A toda mi familia que de una forma u otra siempre me han dado ánimo a seguir con mis metas.

A mi novio por soportarme y estar conmigo en los buenos y malos momentos, por ser paciente y darme todo su amor y su cariño.

A mis amigas Beliza y Yenmy, que desde que las conozco siempre han sido mis verdaderas amigas, ellas son especiales y únicas para mí.

A mis tutores por su apoyo, a Miguel Jaeger que fue de gran ayuda.

Al grupo 3501, por tantos momentos lindos, en especial a Luis Ernesto y a Josué por ayudarme cuando más lo necesitaba.

A Mayler por ser mi amiga en las buenas y en las malas.

A Juliet que además de ser mi compañera de tesis, ha sido una muy buena amiga.

A la Revolución por haberme dado la oportunidad de estudiar en esta gran universidad, en la que he vivido una de las mejores etapas de mi juventud.

A todas las personas que de una forma u otra contribuyeron a que me formara como profesional.

A todos, Gracias.

De Juliet:

A mi hermana, a quien quiero con mi vida, pues ha sido como una madre para mi desde que era pequeña, apoyándome y ayudándome siempre lo mejor que ha podido.

A mis padres por darme la vida y por haberme impulsado a estudiar y llegar hasta aquí.

A mi cuñado que es como mi hermano mayor, y se ha preocupado siempre por mí.

A mi abuelita Eneyda que sé que está muy orgullosa de mi y aunque no esté aquí conmigo se que estoy en su pensamiento en este momento.

A mis tíos, que siempre me han querido mucho y guiado en todo momento, en especial a José Alberto, Armando, Andrés y Roberto.

A los amigos que he tenido en todo este tiempo, a Robin, Eloy, Franke, Freddy,J orgito, Dayana.

A mi ¨pikete¨ favorito: Henry, Daniel, Tomás, Ayata, Carlos o ¨Carlistony¨ y Félix, que lamento no haberlos conocido antes pues me han enseñado a quererlos y a apreciar su amistad; además de haberme contagiado con su espíritu loco y fiestero. Les doy gracias, ya que por ustedes conocí a mi novio, el cual se ha convertido en poco tiempo en una persona muy especial. Su amor ha sido fundamental para salir adelante estos últimos meses.

A Mayler ¨la caimana¨ que se ha convertido en una buena amiga para mí y compañera de parrandas.

A Daylín ¨la negación¨ que no solo es mi compañera de tesis sino que es una gran amiga ante todo.

A mis compañeros de aula, en especial: Josué y Yosviel que muchas veces me ayudaron a estudiar para los exámenes.

A mis tutores, a Miguel que nos ayudó mucho.

A la Revolución por haberme dado la oportunidad de estudiar en esta grandiosa escuela en la cual he vivido una de las mejores etapas de mi vida.

A todas las personas que de una forma u otra pusieron su granito de arena para que todo saliera bien.

Dedicatoria

De Daylín:

A mis padres por todo el esfuerzo y sacrificio en aras de que pudiera realizar mis sueños.

De Juliet:

A mis padres y a mi hermana.

Resumen

En los últimos años se han disparado las opciones para automatizar el control de la calidad de software. Han pasado los años en que se hacían inspecciones de código a mano y por muestreo, a multitud de herramientas de calidad software que contienen distintos tipos de prueba que permiten asegurar la calidad del software creado. La presente investigación está asociada al Centro Nacional de Calidad de Software (CALISOFT) de la Universidad de las Ciencias Informáticas (UCI), el cual se dedica a realizar diferentes pruebas para evaluar la calidad de los productos de software antes de ser liberados. Este centro actualmente no cuenta con una herramienta que permita realizar este proceso sino que lo efectúa manualmente y esto trae muchas desventajas para el mismo. En este trabajo se presenta un subsistema que permitirá gestionar de forma automatizada el proceso de gestión de métricas de calidad para la evaluación de software, fortaleciendo este proceso en gran manera. Además propone un sistema de métricas guiados por las características de calidad según el modelo de calidad que plantea la ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad para las pruebas desarrolladas en CALISOFT a los productos de los proyectos desarrollados en la UCI. Se describen los diferentes conceptos y definiciones esenciales para una mejor comprensión del tema como es el caso de medida, medición y métrica. Para facilitar la implementación del sistema propuesto se hará uso de un Sistema Gestor de Contenidos (CMS) Drupal junto a otras herramientas compatibles con este, que facilitan el desarrollo de la aplicación.

Palabras Claves
Calidad, Métricas, Prueba de liberación

Índice

Introducción

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

Nuestro país ha visto en las TIC una vía para el desarrollo económico y cultural, tomando como iniciativa llevar un proceso de informatización en la sociedad cubana. Uno de los principales pasos de este proceso fue la creación de los Joven Club de Computación y Electrónica, y la fundación de la Universidad de las Ciencias Informáticas, en la cual existen varios centros que brindan servicios de desarrollo de software, entre los que se encuentra CALISOFT[1].

En el centro CALISOFT se realizan mediciones para comprobar si los artefactos liberados cumplen con las métricas de calidad establecidas, las mismas son realizadas de forma manual, lo que conlleva a un retraso e incide negativamente en la calidad de las pruebas de liberación y en la satisfacción del cliente. No se han identificado todas las métricas necesarias y por consiguiente no se gestionan; al no gestionarse dichas métricas no se evalúan los atributos de calidad relacionados a ellas, que están establecidos según el modelo de calidad ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad. Actualmente el cálculo de las métricas es muy tedioso y consume mucho tiempo de los especialistas al frente de las pruebas. Por otro lado, existen dificultades en la toma de decisiones pues los directivos de la universidad no tienen como argumentar de una forma eficiente el estado en que se encuentran los proyectos en el proceso de las pruebas de liberación.

Todo lo antes descrito da lugar al siguiente **Problema a resolver:** ¿Cómo contribuir a la gestión de las métricas de calidad para agilizar el tiempo y el almacenamiento de los datos en el proceso de las pruebas de liberación que se realizan en CALISOFT? Para esto se plantea el siguiente **Objeto de estudio:** Proceso de pruebas de liberación, teniendo como **Objetivo general:** Desarrollar una herramienta informática para la gestión de las métricas de calidad que permita agilizar el tiempo y el almacenamiento

[1] Centro Nacional de Calidad de Software

de los datos en el proceso de pruebas de liberación en CALISOFT y como **Campo de acción:** Herramienta de gestión de métricas de calidad.

Como **resultado** de esta investigación se obtendrá una herramienta informática para la gestión de las métricas de calidad que agilice el tiempo y el almacenamiento de los datos en el proceso de las pruebas de liberación en CALISOFT.

Se tiene como **Idea a defender:** Si se desarrolla una herramienta informática para la gestión de las métricas de calidad entonces se agiliza el tiempo y el almacenamiento de los datos en el proceso de pruebas de liberación en CALISOFT.

Se tienen como **Objetivos específicos:**

1. Realizar la fundamentación teórica para la selección de las herramientas y demás complementos a utilizar.
2. Diseñar la herramienta informática para la gestión de métricas de calidad en las pruebas de liberación.
3. Implementar la herramienta informática para la gestión de métricas de calidad en las pruebas de liberación.
4. Aplicar pruebas a la herramienta informática para verificar el cumplimiento de los requisitos funcionales y no funcionales.

Para dar cumplimiento a este trabajo se definen las siguientes **Tareas de la investigación:**

1. Revisión de las tendencias actuales para la gestión de un sistema de métricas de calidad en las pruebas de liberación.
2. Análisis del proceso de gestión de las métricas de calidad en las pruebas de liberación en CALISOFT, atendiendo al estado del arte y las condiciones propias del centro.
3. Análisis de herramientas que gestionen las métricas en las pruebas para la gestión de un sistema de métricas de calidad en las pruebas de liberación.
4. Entrevistas con los especialistas de calidad del centro CALISOFT para obtener más información sobre el tema.

5. Realización de un estudio de las metodologías de desarrollo de software para realizar el análisis y el diseño de la herramienta.
6. Análisis de las tecnologías y herramientas a utilizar para la implementación del sistema.
7. Selección de las tecnologías y herramientas a utilizar para la implementación del sistema.
8. Análisis de la herramienta para la gestión de las métricas en las pruebas de liberación.
9. Diseño de la herramienta para la gestión de las métricas en las pruebas de liberación.
10. Implementación de la herramienta para la gestión de las métricas en las pruebas de liberación.
11. Realización de las pruebas de software para verificar el cumplimiento de los requisitos funcionales y no funcionales.

En la presente investigación se hace uso de los siguientes **Métodos científicos:**

Métodos Teóricos: Permiten estudiar las características del objeto de investigación que no son observables directamente.

- Histórico-Lógico: para el estudio crítico de los trabajos anteriores, y para utilizar estos como punto de referencia y comprobación de los resultados alcanzados.
- Analítico-sintético: para realizar un estudio de los diferentes documentos y bibliografías relacionadas con el tema, para luego extraer los elementos más importantes que permitan comprender qué son las métricas de calidad y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta.
- Modelación: para la realización de la investigación se apoyará en modelos existentes para lograr un mejor entendimiento de la misma.

Métodos Empíricos: Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional.

- Entrevista: para conocer cómo se desarrolla el proceso de gestión de métricas de calidad en los proyectos productivos y precisar en qué medida se retroalimentan los directivos de la universidad del resultado de las pruebas que se realizan en el departamento de pruebas de software.

- Medición: para obtener información numérica basándose en métricas que permitan medir la calidad del software.

Este documento contiene tres capítulos donde se describe todo el proceso de la investigación.

- ¡Error! No se encuentra el origen de la referencia.**:** contiene los conceptos y fundamentos necesarios para el dominio del problema y comprensión del resto del documento, así como las tecnologías y herramientas necesarias para la implementación del sistema.
- **Capítulo 2: Características y análisis del sistema:** realiza el análisis del negocio, definiendo los actores y roles con sus respectivas responsabilidades, la captura de requisitos, las historias de usuario y el plan de release. Además se define la estructura de la aplicación mediante el diagrama de modelo de dominio, el diagrama de paquetes y el diagrama de base de datos.
- **Capítulo 3: Implementación y prueba:** contiene la implementación del sistema y modelo de prueba del software para garantizar el cumplimiento de los requisitos del cliente.

Capítulo 1.Fundamentación teórica

En este epígrafe se verán las principales características de los procesos de medición y su definición para lograr una mejor comprensión de la propuesta en cuestión. Se describen las métricas de software y los distintos niveles organizacionales, así como los estándares y modelos que se utilizan en la implementación de las métricas. Se verán además las tendencias actuales en cuanto a métricas de calidad y las herramientas para la gestión de las mismas tanto a nivel mundial, como en el país y en la UCI.

1.1 Estado del arte

La medición es fundamental para cualquier disciplina de ingeniería, y la ingeniería del software no es una excepción. La medición nos permite tener una visión más profunda proporcionando un mecanismo para la evaluación objetiva. Lord Kelvin en una ocasión dijo: "*Cuando pueda medir lo que está diciendo y expresarlo con números, ya conoce algo sobre ello; cuando no pueda medir, cuando no pueda expresar lo que dice con números, su conocimiento es precario y deficiente: puede ser el comienzo del conocimiento, pero en sus pensamientos, apenas está avanzando hacia el escenario de la ciencia*".

Con el objetivo de desarrollar la investigación de manera satisfactoria primeramente se verán los conceptos y definiciones asociados al tema en cuestión y que sirve como punto de partida para el desarrollo de la solución propuesta.

1.1.1 Calidad de software

El objetivo de todo proceso de desarrollo de software es obtener como resultado un producto con calidad y que satisfaga las necesidades del cliente.

La calidad de software puede definirse como concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente (Pressman, 2005).

1.1.2 Ventajas de implantar modelos o estándares de calidad de software:

- Tener una oportunidad para corregir los procesos de software que se hayan desajustado con el tiempo. Cuando se usa un modelo de calidad se realiza un seguimiento al proceso de desarrollo, esto permite que se puedan hacer correcciones en un momento dado y asegurar la obtención de un buen producto.
- Clasificar a las empresas como de clase mundial. Las empresas tienden a ser las mejores en su campo. El uso de modelos de calidad estándar les permite diferenciar las empresas de muchas otras que no tienen un modelo establecido para evaluar la calidad de sus productos.
- Lograr que la empresa de software sea más competitiva.
- Certificar la competitividad internacional requerida para competir en todos los mercados. El uso adecuado de los modelos de calidad permite estar a la vanguardia de las mejores empresas, haciendo que crezca su competitividad en el medio.
- Cambiar la actitud del personal de la empresa.
- Desarrollar y mejorar el nivel y la calidad de vida del personal. Al aplicar un modelo de calidad de software las personas tienen un proceso de desarrollo/evaluación definido y claro que facilita las labores.
- Realizar una mejora continua en la calidad de los procesos de software utilizados, servicios y productos de software.
- Reducir los costos en todos los procesos. Al aplicar un modelo de calidad los procesos se vuelven más organizados, lo que repercute en la agilización de los mismos y genera reducción de costos.
- Aumentar la productividad, efectividad y utilidad de la empresa. El adoptar un modelo de calidad permite tener un proceso definido y bien documentado, hay una definición y comprensión de los roles de trabajo y hay una coherencia con el trabajo que se está realizando.
- Asegurar la satisfacción de los clientes internos y externos.
- Tener criterios de medición e indicadores congruentes que se utilizan en la empresa para comparar respecto de las mejores prácticas, para conocer fortalezas y debilidades de la empresa y establecer las estrategias necesarias para realizar mejoras. Al tener un modelo de calidad, las mediciones que salgan a partir del uso de este sirven como histórico para otros futuros productos software desarrollados (Mulén Mustelier, y otros, 2010).

1.2 Modelos de Calidad

Entre los modelos de calidad existentes el más actual es la norma ISO/IEC 25000, la cual proporciona una guía para el uso de las nuevas series de estándares internacionales llamados Requisitos y Evaluación de Calidad de Productos Software, sin embargo tiene la desventaja de que sus métricas están incompletas, por lo cual, para el desarrollo de la investigación se utilizará la norma ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad, ya que es de los más usados por empresas cubanas para la definición de métricas. Además muchas características de este modelo hacen referencia a la operación, transición y revisión del software. También es muy flexible y fácil de implementar, pues provee a las empresas un entorno para que lo adecuen a sus necesidades y definan su propio modelo de calidad.

La norma ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad se basa en que el objetivo no es necesariamente alcanzar una calidad perfecta, sino la necesaria y suficiente para cada contexto de uso a la hora de la entrega y del uso del software por parte de los usuarios y es necesario comprender las necesidades reales de los usuarios con tanto detalle como sea posible. Razones por la cual se definen los siguientes aspectos de calidad: interna (medible a partir de las características intrínsecas, como el código fuente), externa (medible en el comportamiento del producto, como en una prueba) y en uso (durante la utilización efectiva por parte del usuario).

La norma ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad permite definir un modelo de calidad, para nuestra organización, en base a las 6 características que se indican en la calidad en el ciclo de vida del software:

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

En cuanto a los factores de calidad de uso la norma ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad reconoce cuatro factores:

- Eficacia.
- Productividad.
- Seguridad.

- Satisfacción.

Se pretende que estas características sean comprensivas, o sea que cualquier componente de la calidad de software se pueda describir como combinación de aspectos de estos factores. A su vez estas características están formadas por sub-características que están refinadas por niveles (Gómez, 2009).
Para cada característica y sub-característica, la capacidad del software es determinada por un conjunto de atributos internos que pueden medirse y se definen de la siguiente manera:

- Funcionalidad: conjunto de atributos que relacionan la existencia de un conjunto defunciones con sus propiedades especificadas. Las funciones satisfacen necesidades especificadas.
- Adecuación: atributos que determinan si el conjunto de funciones son apropiadas para tareas especificadas.
- Exactitud: atributos que determinan que los efectos sean los correctos o los esperados.
- Seguridad: atributos que miden la habilidad para prevenir accesos no autorizados.
- Interoperabilidad: atributos que miden la habilidad de interactuar con sistemas especificados.
- Cumplimiento: atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones legales.
- Confiabilidad: conjunto de atributos que se relacionan con la capacidad del software de mantener su nivel de performance bajo las condiciones establecidas por un período de tiempo.
- Madurez: atributos que se relacionan con la frecuencia de fallas por defectos en el software.
- Tolerancia a las fallas: atributos que miden la habilidad de mantener el nivel especificado de performance en caso de fallas del software.
- Recuperación: atributos que miden la capacidad de restablecer el nivel de performance y recuperar datos en caso de falla, y el tiempo y esfuerzo necesario para ello.
- Usabilidad: conjunto de atributos que se relacionan con el esfuerzo necesario para usar, y en la evaluación individual de tal uso, por parte de un conjunto especificado de usuarios.
- Entendimiento: atributos que miden el esfuerzo del usuario en reconocer el concepto lógico del software y su aplicabilidad.
- Aprendizaje: atributos que miden el esfuerzo del usuario en aprender la aplicación.
- Operabilidad: atributos que miden el esfuerzo de usuario en operar y controlar el sistema.
- Atractivo: atributos que miden la capacidad del producto de software de ser amigable para el usuario.

- Eficacia: conjunto de atributos que se relacionan con el nivel de performance del software y la cantidad de recursos usados bajo las condiciones establecidas.
- En tiempo: atributos que miden la respuesta y tiempos de procesamiento de las funciones.
- En recursos: atributos que miden la cantidad de recursos usados y la duración de tal uso en la ejecución de las funciones.
- Mantenibilidad: conjunto de atributos que se relacionan con el esfuerzo en realizar modificaciones.
- Analizabilidad: atributos que miden el esfuerzo necesario para el diagnóstico de deficiencias, para la identificación de las partes que deben ser modificadas.
- Facilidad para el cambio: atributos que miden el esfuerzo necesario para realizar modificaciones, cambios de fallas en el contexto.
- Estabilidad: atributos que se relacionan con el riesgo de efectos no esperados en las modificaciones.
- Testeabilidad: atributos que miden el esfuerzo necesario para validar el software modificado.
- Portabilidad: conjunto de atributos que se relacionan con la habilidad del software para ser transferido de un ambiente a otro.
- Adaptabilidad: atributos que miden la oportunidad de adaptación a diferentes ambientes sin aplicar otras acciones que no sean las previstas para el propósito del software.
- Instalabilidad: atributos que miden el esfuerzo necesario para instalar el software en el ambiente especificado.
- Conformidad: atributos que miden si el software se adhiere a estándares o convenciones relacionados con portabilidad.
- Reemplazo: atributos que se relacionan con la oportunidad y esfuerzo de usar el software en lugar de otro software en su ambiente (Gómez, 2009).

1.3 Métricas

Aunque los términos medida, medición y métricas se utilizan a menudo indistintamente, es importante destacarlas diferencias sutiles entre ellos. Como los dos primeros pueden ser usados como un nombre o como un verbo, las definiciones de estos se pueden confundir. Dentro del contexto de la ingeniería del software, una medida proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto. 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. El IEEE[2] en su Standard Glossary of Software Engineering Terms (Glosario Estándar de Terminología de Ingeniería de Software) [IEEE93] define métrica como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (Rodríguez, 2011).

1.4 Métricas de Calidad

El principal objetivo de los ingenieros de software es producir sistemas, aplicaciones o productos de alta calidad. Para las evaluaciones que se desean obtener es necesaria la utilización de medidas y técnicas, que evalúan la calidad de manera objetiva. Una métrica de calidad brinda una medida cuantitativa del grado en que un sistema o proceso cumple con los requisitos establecidos por el cliente.

1.5 Métricas de software

Una métrica del software relata de alguna forma las medidas individuales sobre algún aspecto (por ejemplo: el número medio de errores encontrados por revisión o el número medio de errores encontrados por persona y hora en revisiones).

Las métricas son la maduración de una disciplina que ayudan a la evaluación de los modelos de análisis y de diseño proporcionan una indicación de la complejidad de diseños procedimentales y de código fuente, y ayudan en el diseño de pruebas más efectivas (Pressman, 2005).

Pueden definirse además como la aplicación continua de mediciones en el proceso de desarrollo del software y sus productos, para suministrar información relevante a tiempo y contribuir a una mejora tanto del proceso como del producto. Estas comprenden un amplio rango de actividades diversas, algunas de estas son:

- Medidas y modelos de estimación de coste y esfuerzo

[2]Institute of Electrical and Electronics Engineers (Instituto de Ingeniería Eléctrica y Electrónica)

- Aseguramiento y control de calidad
- Modelos de fiabilidad
- Modelos y evaluación de ejecución

1.6 Métricas de Calidad de Software

Los sistemas de métricas de calidad del software tradicionales se han centrado fundamentalmente en las métricas de procesos, de productos y de recursos. Los sistemas de métricas para tiempo de ejecución más comunes hoy en día son los usados en aplicaciones para probar las aplicaciones. Este tipo de aplicaciones usan sistemas de métricas en tiempo de ejecución para medir tiempos, buscar cuellos de botella en las aplicaciones, medir capacidades máximas, etcétera. Así, las métricas tratan de servir de medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento.

Los tres objetivos fundamentales de la medición son:

- Entender qué ocurre durante el desarrollo y el mantenimiento.
- Controlar qué es lo que ocurre en nuestros proyectos.
- Mejorar nuestros procesos y nuestros productos.

La métrica tradicionalmente más relevante de la complejidad del software es la llamada Complejidad ciclomática, que proporciona una medición cuantitativa de la complejidad lógica de un programa como el número de caminos independientes dentro de un fragmento de código. En general los resultados se interpretan dentro de estos rangos: una complejidad ciclomática de 1 a 10 es un programa simple sin mucho riesgo; de 10 a 20 es un riesgo más complejo; de 21 a 50, muy complejo, un programa de alto riesgo y más de 50, programa no <<testeable>> (ya que la métrica indica el número mínimo de caminos independientes que han de ejecutarse (tests) para estar seguros de haber ejecutado al menos una vez todas las sentencias de un programa y todas las condiciones lógicas en sus vertientes verdadera y falsa).

Las métricas para sistemas orientados a objetos deben de ajustarse a las características que distinguen el software orientado a objetos del software convencional. Estas métricas hacen hincapié en el encapsulamiento, la herencia, complejidad de clases y polimorfismo. Por lo tanto las métricas orientadas a objetos se centran en métricas que se pueden aplicar a las características de encapsulamiento,

ocultamiento de información, herencia y técnicas de abstracción de objetos que hagan única a una clase concreta (García Sánchez, 2010).

1.6.1 Clasificación de las métricas de software

Las métricas pueden ser agrupadas teniendo en cuenta diferentes factores:

- Complejidad: definen la medición de la complejidad, volumen, tamaño, anidaciones, y configuración.
- Calidad: definen la calidad del software en cuanto a exactitud, estructuración o modularidad, pruebas, mantenimiento.
- Competencia: intentan valorar o medir las actividades de productividad de los programadores con respecto a su certeza, rapidez, eficiencia y competencia.
- Desempeño: miden la conducta de módulos y sistemas de un software, bajo la supervisión del sistema operativo (SO) o hardware.
- Estilizadas: de experimentación y preferencia, estilo de código, convenciones, limitaciones.

Según Pressman existen categorías o grupos de métricas distintos:

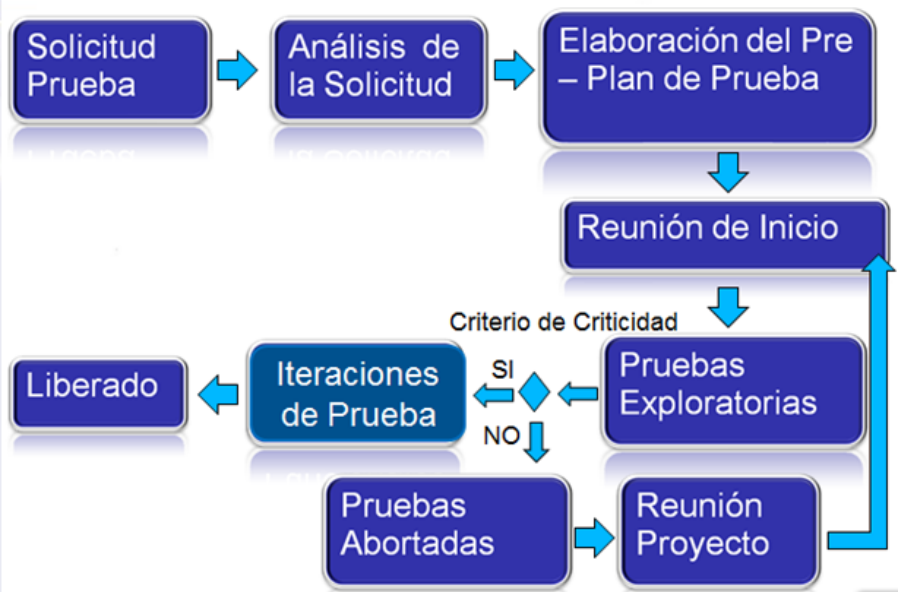
- Métricas técnicas: centradas en las características del software más que en su proceso de desarrollo.
- Métricas de calidad: tanto del software desarrollado como de la efectividad del proceso de la ingeniería aplicado.
- Métricas de productividad: referidas al rendimiento del proceso de desarrollo como función del esfuerzo aplicado.
- Métricas orientadas al tamaño: miden de forma directa el software y el proceso por el cual se desarrolla.
- Métricas orientadas a la función: centradas en la funcionalidad o utilidad del programa.
- Métricas orientadas a la persona: aportan información sobre la forma en que la gente desarrolla software.

De las métricas abordadas anteriormente se utilizarán el desarrollo del sistema solo las métricas de calidad.

1.7 Procedimiento para utilizar el catálogo de métricas en la pruebas del LIPS

El proceso de pruebas de liberación en el Laboratorio Industrial de Pruebas de Software (LIPS) comienza con una solicitud de prueba, luego se realiza el análisis de la solicitud, donde el jefe de laboratorio adiciona el proyecto, centro, artefactos y los especialistas de calidad que van a hacer el tipo de prueba. Se elabora el pre- plan de prueba, el cual lo realiza el especialista de calidad; en este aparecen los tipos de pruebas que se van a aplicar y las métricas asociadas a estas. Después se realiza una reunión de inicio en la cual participan los miembros del equipo de desarrollo y los especialistas de calidad que guiarán las pruebas, al ser terminada, tienen que quedar aprobados los tipos de pruebas que se van a realizar y las métricas a aplicar (ver anexo 3), en la cual el especialista de calidad, al cual le fue dado el proyecto con antelación, realiza la asignación de las métricas a los artefactos en la aplicación. Más adelante el especialista de calidad procede a realizar las pruebas exploratorias, en las cuales, si el producto cumple con los criterios de criticidad, se le aplican las pruebas definidas anteriormente, de lo contrario se abortaría el proceso de pruebas. Se realizan las iteraciones de pruebas al producto, en las que se obtienen los datos para poder calcular las métricas seleccionadas anteriormente. Al terminarse este proceso se libera el producto.

A continuación se muestra un esquema representando el proceso descrito anteriormente:



Esquema 1. Proceso de pruebas de liberación.

1.8 Herramientas para la gestión de las métricas de calidad

En la actualidad es una práctica muy difundida en el mundo de las TICs que las empresas que brindan servicios de calidad cuenten con herramientas y sitios web que posibilitan dar a conocer los productos y servicios que ofrecen, además de brindarlos con mayor calidad.

Una de las compañías líderes y más exitosas en cuanto a prestación de servicios de calidad de software y pruebas, es la SQS[3] S.A. Esta empresa ha alcanzado centrar su objetivo en los servicios de consultoría, programas de formación y desarrollo de nuevas herramientas. SQS enfoca sus servicios de consultoría de calidad de software en mejorar y asegurar la calidad de los procesos de desarrollo de sus clientes y sus productos finales. Entre las herramientas que brindan se encuentran los siguientes:

- SQS TestWORKFLOW.
- SQS AgileREQ.
- SQS Q-Val.
- SQS interCENELEC

1.8.1 SQS TestWORKFLOW

Es una herramienta diseñada para servir como guía en el proceso de validación de software, y facilitar la realización de pruebas y gestión de requisitos de manera automática, y también de la documentación generada en el proceso.

Su concepción y diseño inicial la hacen especialmente indicada para su aplicación en entornos de software embebido en sistemas críticos, como los ferrocarriles, el manejo automático de maquinaria o

[3]Software Quality Systems (Sistema de Calidad de Software)

herramientas. Sin embargo, posee gran flexibilidad y puede ser utilizado en los desarrollos de software en diferentes campos.

Esta herramienta incluye la definición y gestión de requisitos, la organización del proceso de prueba, la ejecución automática de casos de prueba, análisis de resultados y la generación de varios informes. Las relaciones existentes entre los requisitos, las pruebas y las diferentes ejecuciones se registran en todo momento, lo que asegura su trazabilidad durante todo el proceso.

1.8.2 SQS AgileREQ

AgileREQ es una herramienta implementada para la gestión de requisitos. Es especialmente útil en los entornos en los que los cambios en los requisitos son continuos.

Como su nombre indica es una herramienta basada en el proceso de desarrollo siguiendo las metodologías ágiles que pretenden que el desarrollo de un proyecto de software sea un desarrollo adaptativo, aunque disciplinado, y que aporte soluciones sencillas. AgileREQ tiene un enfoque adaptable, en el que la planificación del proyecto progresa a medida que surgen cambios.

1.8.3 SQS Q-Val

Q-Val es la herramienta de apoyo a la metodología desarrollada por SQS con la que se validan aplicaciones web y se evalúa su calidad de una manera sistemática y teniendo en cuenta las características propias de la aplicación y de su proceso de desarrollo. Permite además definir los requisitos (tanto funcionales como no funcionales) que debe cumplir una aplicación web para que pueda ser considerada como un software de calidad y funcional. Es por tanto una ayuda imprescindible para los procesos de validación de aplicaciones web, facilitando el análisis previo a la validación y sirviendo de guía durante el proceso para evitar dejar de lado aspectos importantes.

La base de Q-Val es un completo y muy estructurado conjunto de requisitos que están sometidos a una constante monitorización y actualización. Para la definición de dichos requisitos, se estudian, de forma continua los Modelos, Recomendaciones, Sellos y Estándares de Calidad Web más extendidos.

1.8.4 SQS interCENELEC

Es una herramienta desarrollada para gestionar los desarrollos que deban cumplir con las normas CENELEC 50126, 50128 y 50129. Esta herramienta ofrece al usuario soporte en la evaluación del cumplimiento de dichas normas y proporciona las directrices necesarias durante el proceso de

implantación. Puede ser utilizada tanto por el evaluador externo como por el implantador interno como ayuda a su gestión.

Evaluador externo. Durante la evaluación, la herramienta le ofrece las funcionalidades necesarias para el evaluador externo, tales como:

- Guiarle en su labor de evaluación
- Personalizar sus preguntas
- Gestionar de una manera eficiente la información generada durante la evaluación.

Desde el punto de vista de un equipo o implantador interno, la herramienta cumple un triple propósito:

- Evaluar en qué estado se encuentran sus desarrollos respecto de los niveles de seguridad que le exige la norma.
- Como apoyo durante el proceso de implantación, interCENELEC le indicará las pautas a seguir para alcanzar el nivel de seguridad exigido.
- La posibilidad de poder mostrar a terceros el grado de cumplimiento de la norma (Software Quality Systems S. A., 2013).

1.8.5 Rational Quality Manager Express Edition

Por otra parte IBM[4], que se especializa en el desarrollo de herramientas software, desarrolló el **Rational Quality Manager Express Edition** (Edición Rápida de Manejador de Calidad Racional), herramienta para la gestión de pruebas y gestión de la calidad para ofrecer software orientado a la calidad.

Rational Quality Manager Express Edition ha sido diseñado para crear sinergia en los esfuerzos de un equipo dedicado a crear software, ayudando a reducir los defectos y acelerar los programas de entrega del producto y para la colaboración entre pequeñas y medianas empresas, ya que les permite compartir información, usar la automatización para agilizar las programaciones de proyectos y generar informes de mediciones sobre los proyectos con el fin de tomar decisiones bien fundamentadas. Además contribuye a que tengan un mejor control de sus proyectos mediante la provisión de métricas confiables y oportunas. También soporta gran cantidad de funciones de usuario como: gerente de pruebas, arquitecto de pruebas, líder de pruebas y gerente de laboratorio (IBM, 2008).

La tabla que se muestra a continuación, evidencia una comparación entre las herramientas antes explicadas para gestionar las métricas de calidad en cuanto a: costo, funcionalidades que desea CALISOFT, adaptabilidad al flujo de trabajo de CALISOFT y licencia.

Herramientas existentes en el mundo	Costo	Funcionalidades que desea CALISOFT	Adaptabilidad al flujo de trabajo de CALISOFT	Licencia
SQS TestWORKFLOW.	Muy costosa	Incompletas	No se adapta	Privativa
SQS AgileREQ	Muy costosa	Incompletas	No se adapta	Privativa
SQS Q-Val	Muy costosa	Incompletas	No se adapta	Privativa
SQS interCENELEC	Muy costosa	Incompletas	No se adapta	Privativa
Rational Quality	Muy costosa	Incompletas	No se adapta	Privativa

[4] International Business Machines (Máquina Internacional de Negocio)

Manager Express Edition				

Tabla 1. Criterios de comparación sobre algunas herramientas existentes en el mundo respecto a CALISOFT.

Son muchas las herramientas desarrolladas con el fin de facilitar el proceso de pruebas de software y la gestión de las métricas de calidad que rigen el mismo, puesto que resulta un proceso largo y engorroso. Cada una de las herramientas antes mencionadas, posibilitan y agilizan la gestión de las métricas de calidad, algunas son aplicaciones de escritorio, otras son aplicaciones web, sin embargo tienen un inconveniente en común y es que para hacer uso de dichas aplicaciones es necesario abonar un precio, y ¿por qué comprar un producto, bajo el riesgo de que en algún momento aparezca alguna funcionalidad que no pueda satisfacer, aunque sea muy pequeña, si es posible crear una ajustada a las necesidades particulares de una organización, con esfuerzos y gastos mínimos? Por esta razón se ha decidido crear una herramienta para la gestión de las métricas de calidad basadas en el modelo ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad según la propuesta de solución.

1.9 Tecnologías y herramientas para el desarrollo del software

Las tecnologías han influido de diferentes formas en el contexto humano y ha provocado un cambio drástico en la misma. A través de éstas se pueden desarrollar más fácilmente los proyectos que se realizan, ya que se obtiene un resultado más confiable y en el menor tiempo posible. A continuación se dará una breve descripción de las diferentes tecnologías y herramientas que se utilizarán en el desarrollo de la aplicación.

1.9.1 Metodologías ágiles

Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste. Las metodologías ágiles presentan diversas ventajas, entre las que se pueden destacar:

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos breves de software funcional.

- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Importancia de la simplicidad, eliminado el trabajo innecesario.
- Atención continua a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo (Amaro Calderón, et al., 2007).

Para el desarrollo de la aplicación se utilizarán las metodologías ágiles debido a que se adaptan a las necesidades del cliente, ya que es un proyecto de corta duración. Elimina el trabajo innecesario y posee capacidad de realizar cambios a lo largo del desarrollo.

1.9.1.1 SXP

SXP es un híbrido de metodologías ágiles que tiene como base las metodologías SCRUM y XP que permiten actualizar los procesos de desarrollo de software para el mejoramiento de su producción. Esta metodología ayuda a fortalecer el trabajo en equipo, enfocados en una misma dirección, permitiendo además seguir de forma clara el avance de las tareas a realizar, a partir de la inserción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la producción, aumentando el nivel de interés del equipo.

Consta de 4 fases: **Planificación-Definición**, **Desarrollo**, **Entrega** y **Mantenimiento**, cada una desglosada en flujos de trabajo y actividades que generan artefactos.

- **Planificación-Definición:** en esta fase se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- **Desarrollo:** se realiza la implementación del sistema y las pruebas al mismo para comprobar que esté listo para ser entregado.
- **Entrega:** entrega de la documentación.
- **Mantenimiento:** donde se realiza el soporte para el cliente.

De ellas se despliegan 7 flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implementación, prueba, entrega de la documentación, soporte e investigación, el cual se utiliza por el equipo de desarrollo cuando sea necesario, es decir, es un flujo que se puede mover y utilizarlo en cualquier parte del ciclo de vida del proyecto.

De estos flujos se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de reserva del producto, definición de las Historias de usuario, implementación,

planificación de las iteraciones y las actividades que se van a realizar para lograr el producto, pruebas, además de las tareas necesarias para realizar las investigaciones para documentar todo el proceso (Romero, et al., 2010).

Se utilizará para el desarrollo de la aplicación la metodología SXP porque está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Además presenta simplicidad en las soluciones implementadas. Debido a que el sistema debe desarrollarse en poco tiempo y no es necesaria una extensa documentación para su desarrollo.

1.9.2 Sistema de Gestión de Contenido (CMS)

Los sistemas de gestión de contenidos (Content Management Systems o CMS) es un software que se utiliza principalmente para facilitar la gestión de webs, ya sea en Internet o en una intranet, y por eso también son conocidos como gestores de contenido web (Web Content Management o WCM). Hay que tener en cuenta, sin embargo, que la aplicación de los CMS no se limita sólo a las webs.

1.9.2.1 Drupal 7.21

Drupal es un sistema de gestión de contenido modular multipropósito y muy configurable que permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web. Es un programa libre, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema (Cuerda, 2004).

El CMS Drupal se utilizará para el desarrollo de la aplicación ya que se puede descargar y usar sin costo alguno, se puede acceder al código, modificarlo, mejorarlo o adaptarlo a sus necesidades. Además el sistema gestiona todos los detalles técnicos y administrativos e incluye además un conjunto de módulos que pueden reutilizarse en el desarrollo del sistema, lo cual reduciría el tiempo de desarrollo del mismo.

1.9.3 Lenguaje de programación PHP 5.4.3

El PHP (acrónimo de PHP: Hypertext Preprocessor), es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. El PHP inició como una modificación a Perl escrita por Rasmus Lerdorf a finales de 1994. Su primer uso fue el de mantener un control sobre quien visitaba su currículum en su web.

"El PHP es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas" (Henst, 2001).

Se utilizará el lenguaje PHP para la implementación del lado del servidor de los servicios necesarios para la gestión de la información del sistema; además es el lenguaje que utiliza el CMS Drupal, con el cual se realizará la aplicación.

1.9.4 NetBeans 7.2

NetBeans IDE[5] permite desarrollar rápida y fácilmente aplicaciones de escritorio en Java, aplicaciones para móviles y web, mientras que también proporciona una gran herramienta para los lenguajes PHP y C / C + +. Se puede instalar en todos los sistemas operativos que soportan Java, desde Windows a Linux para sistemas Mac OS. Es gratuito y de código abierto y tiene una gran comunidad de usuarios y desarrolladores de todo el mundo.
NetBeans IDE proporciona soporte de primera clase completo para las últimas tecnologías Java y las mejoras más recientes de Java antes de otros IDE (Oracle Corporation, 2012) .

Dado que el sistema a implementar estará necesariamente constituido por páginas JavaScript, php, html, y ccs para la creación de las interfaces de usuario y las clases controladoras del lado del servidor; se utilizará el IDE de desarrollo NetBeans para editar dichas páginas de los módulos creados en el CMS Drupal; con el objetivo de personalizarlas y adaptarlas al ambiente de trabajo y además crear las páginas extras necesarias, tanto del lado del cliente como del servidor, que no sean generadas por el CMS y sirva

[5] Integrated Development Environment (Entorno Integrado de Desarrollo)

de apoyo al sistema en general que complementen las páginas de los módulos generados, o en caso de que se necesiten implementar nuevos módulos que no estén definidos en el CMS a utilizar.

1.9.5 PostgreSQL 8.4

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. Utiliza un modelo cliente/servidor. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Seguidamente se muestra una imagen con los componentes más importantes en un sistema PostgreSQL:

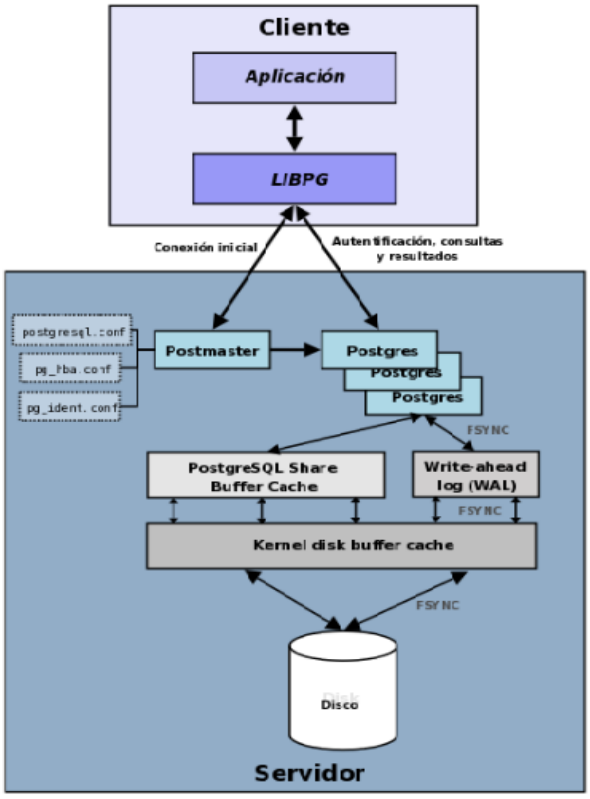


Figura 1. Componentes más importantes en un sistema PostgreSQL.

- **Aplicación cliente:** Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP o sockets locales.

- **Demonio postmaster:** Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargarán de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes.
- **Ficheros de configuración:** Los 3 ficheros principales de configuración utilizados por PostgreSQL, postgresql.conf, pg_hba.conf y pg_ident.conf.
- **Procesos hijos postgres:** Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.
- **PostgreSQL share buffer cache:** Memoria compartida usada por PostgreSQL para almacenar datos en caché.
- **Write-Ahead Log (WAL):** Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO).
- **Kernel disk buffer cache:** Caché de disco del sistema operativo.
- **Disco:** Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione (Martinez, 2010).

Se utilizará PostgreSQL porque es el gestor de base de datos que utiliza Drupal por defecto siendo uno de los más estables y soportado por el CMS. Además si uno de los procesos falla no afectará el resto y el sistema continuará funcionando.

1.9.6 Servidor Apache 2.2.22

Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh, entre otras. Es el servidor web más utilizado mundialmente. El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.
Apache es quizás el servidor web más excelente, por su adaptación y robustez, siendo además casi universal al existir versiones para una multitud de Sistemas Operativos.

Trabaja con gran cantidad de lenguajes, entre ellos PHP y otros lenguajes de script, Java y páginas JSP, teniendo todo el soporte que se necesita para tener páginas dinámicas.

Se utilizará el Servidor Apache para el desarrollo de la aplicación pues es compatible con el gestor de base de datos a utilizar. Además trabaja con el lenguaje PHP y posee todo el soporte que se necesita para implementar páginas dinámicas. Dado que la versión 2.2.22 de Apache, soporta el lenguaje PHP en

su versión 5.4.3, es el idóneo para montar el CMS Drupal 7.21, el cual es el seleccionado, por su estabilidad, para el desarrollo del sistema y este CMS requiere dicha versión para PHP.

1.9.7 Visual Paradigm 8.0

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos.

Visual Paradigm también ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador automáticamente de layout.
- Sincronización de código fuente en tiempo real (Paradigm, 2012).

Se utilizará el Visual Paradigm para el modelado UML del desarrollo y el negocio del sistema. Para la representación de algunos artefactos, lograr estabilidad y servir de guía durante el desarrollo de la aplicación.

1.10 Conclusiones del capítulo

- Se obtuvo mediante elementos teóricos, una base de conocimientos elementales de la investigación, a través de definiciones, conceptos y modelos que son el soporte del estudio realizado.
- El proceso de desarrollo del sistema estará regido por el lenguaje, metodología y herramientas propuestas para la implementación de la herramienta para la gestión de métricas de calidad en las pruebas de software.

Capítulo 2. Características y análisis del sistema

En este capítulo se dará una breve descripción del rol que desarrolla el proceso de gestión de las métricas, así como el papel que juegan otras personas que puedan estar involucradas. Se definirá además la información manejada y la propuesta del sistema. Se realiza una descripción de la solución propuesta, mediante los diagramas de procesos correspondientes al sistema. Se expone el listado de requisitos funcionales y no funcionales, historias de usuarios y la definición de los escenarios, el diagrama del modelo de dominio, el diagrama de despliegue, el diagrama de paquetes y el de base de datos. Además se describe la arquitectura propuesta para el módulo y patrones de diseño utilizados para la implementación del mismo.

2.1 Propuesta del sistema

Para contribuir a la gestión de las métricas se implementó una herramienta que interviene en el proceso de recolección de los datos de las mismas, ayudando a los especialistas de calidad al frente de las pruebas a hacer este proceso lo más rápido y menos complejo posible. Realiza los cálculos de todas las métricas definidas ya que en ocasiones el cálculo de las fórmulas de las estas es complicado y retrasan los diferentes procesos que se estén realizando. También informatiza algunas funciones básicas del proceso de gestión de las métricas.

2.2 Lista de reserva del producto

La Lista de reserva del producto (LRP), es uno de los artefactos generados en la etapa de captura de requisitos, está conformada por una lista priorizada que define los requerimientos técnicos y del negocio, funciones, actualizaciones y tecnológicas requeridas. A continuación se muestra la Lista de reserva del producto correspondiente a la herramienta para la gestión de métricas de calidad en las pruebas de software.

Asignado	Ítem	Descripción	Estimación	Estimado por
Muy Alta				
Juliet	1	Adicionar proyecto	3	Analista

Daylín	2	Insertar datos de las métricas	3	Analista
Juliet	3	Calcular métricas	4	Analista
Daylín	4	Generar reportes de las métricas por proyecto	2	Analista
Juliet	5	Imprimir Reportes de las métricas calculadas por proyectos	2	Analista
Daylín	6	Adicionar tipo de proyecto	3	Analista
Juliet	7	Adicionar Centro	3	Analista
Daylín	8	Adicionar tipo de artefacto	2	Analista
Daylín	9	Adicionar iteración	3	Analista
Juliet	10	Autenticar Usuario	4	Analista
Daylín	11	Adicionar roles	4	Analista
Juliet	12	Adicionar usuario	3	Analista
Daylín	13	Asignar métricas a un artefacto	3	Analista
Alta				
Juliet	14	Modificar proyecto	1	Analista
Daylín	15	Modificar tipo de proyecto	1	Analista
	16	Modificar Centro	1	Analista
Daylín	17	Modificar tipo de artefacto	1	Analista
Juliet	18	Modificar usuario	1	Analista
Juliet	19	Modificar iteración	1	Analista
Daylín	20	Modificar roles	1	Analista
Media				
Juliet	21	Eliminar proyecto	1	Analista
Daylín	22	Eliminar asignación de métricas a un artefacto	1	Analista

Juliet	23	Eliminar tipo de proyecto	1	Analista
Daylín	24	Eliminar Centro	1	Analista
Juliet	25	Buscar Centro	1	Analista
Daylín	26	Buscar tipo de proyecto	1	Analista
Juliet	27	Buscar métrica asignada	2	Analista
Daylín	28	Buscar Proyecto	2	Analista
Juliet	29	Eliminar tipo de artefacto	1	Analista
Daylín	30	Buscar tipo de artefacto	1	Analista
Juliet	31	Eliminar iteración	2	Analista
Daylín	32	Buscar iteración	2	Analista
Juliet	33	Eliminar roles	2	Analista
Daylín	34	Eliminar usuario	1	Analista
Baja				
Daylín	35	Visualizar proyecto	3	Analista
Juliet	36	Visualizar asignación de métricas a un artefacto	3	Analista
Daylín	37	Visualizar cálculo de las métricas	4	Analista
Juliet	38	Visualizar tipo de proyecto	3	Analista
Daylín	39	Visualizar Centro	2	Analista
Juliet	40	Visualizar tipo de artefacto	2	Analista
Juliet	41	Visualizar iteración	3	Analista
Daylín	42	Visualizar roles	2	Analista
Juliet	43	Visualizar Usuario	2	Analista
Requisitos no Funcionales				
Software				
Daylín	44	Sistema operativo a utilizar:		Analista

		Multiplataforma		
Juliet	45	Sistema Gestor de Bases de Datos: • PostgreSQL 5.5.24		Analista
Hardware				
Daylín	46	Servidor con al menos: 2 GB RAM, Core 2 due, HDD (160-250 GB).		Analista
Portabilidad				
Juliet	47	El producto debe cumplir con los requisitos de integración con el resto de las aplicaciones desarrolladas o por desarrollar en CALISOFT		Analista
Seguridad				
Daylín	48	Los permisos de accesibilidad, lectura y escritura son gestionados en dependencia del rol asignado a cada usuario del sistema.		Analista
Legales				
Juliet	49	Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de software libre.		Analista
Eficiencia				
Daylín	50	• Soportar 60 a 70 usuarios conectados. • Soportar 10 a 15 usuarios trabajando en la misma funcionalidad.		Analista
Ayuda y documentación en línea				
Daylín	51	Como evidencia de la evolución del Proyecto se entregarán los siguientes documentos: • Historias de Usuarios • Casos de Prueba de Funcionalidad • Tareas de ingeniería • Plan de release • Plan de prueba		Analista

Tabla 2. Lista de reserva del producto (LRP).

2.3 Validación de requisitos

Es importante asegurar la validez de los requisitos previamente a comenzar un desarrollo de software. Para ello debe de hacerse una comprobación de la correspondencia entre las descripciones iniciales y si

el modelo es capaz de responder al planteamiento inicial. Para llevar a cabo esto, se suele realizar comprobando que el modelo obtenido responde de la misma forma deseada que la que el cliente pide por una parte, y por otra a la inversa, si otras respuestas del modelo convencen al cliente. En algunos casos será necesario construir prototipos con una funcionalidad similar muy reducida para que el cliente se haga una idea aproximada del resultado.

La validación de los requisitos, obviamente tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación con el costo que eso conlleva. Los parámetros a validar en los requisitos son:

- **Validez:** No basta con preguntar a un usuario, todos los potenciales usuarios pueden tener puntos de vista distintos y necesitar otros requisitos.
- **Consistencia:** No debe haber contradicciones entre unos requisitos y otros.
- **Completitud:** Deben estar todos los requisitos. Esto es imposible en un desarrollo iterativo, pero, al menos, deben estar disponibles todos los requisitos de la iteración en curso.
- **Realismo:** Se pueden implementar con la tecnología actual.
- **Verificabilidad:** Tiene que existir alguna forma de comprobar que cada requisito se cumple (MADEJA, 2013).

2.3.1. Validación de los requisitos del sistema a implementar

De los métodos más usados para validación de requisitos se le aplicarán al sistema los siguientes: Revisiones de Requisitos y el uso de la Métrica para la calidad de la especificación.

2.3.1.1 Revisiones de requisitos

El método de revisiones de requisitos consiste en realizar una o varias reuniones donde un equipo de trabajo intenta localizar errores en la definición de los requisitos. La revisión de requisitos es uno de los mejores métodos de validación de requisitos. Generalmente hablando, las revisiones de requisitos permiten:

- Descubrir una gran cantidad de defectos en los requisitos.
- Reducir los costos de desarrollo entre un 25% y un 30%.
- Reducir el tiempo de pruebas entre un 50% y un 90%.

El resultado final de las reuniones de revisión es un documento que contiene la lista de defectos localizados y una lista de acciones recomendadas (Master de Ingeniería de Software-A Distancia, 2012).

El equipo de trabajo que realizó las diferentes reuniones para la validación de los requisitos del sistema estuvo compuesto por:

1. MSc. Asnier Enrique Góngora Rodríguez
2. Ing. Lisandra Díaz Figueredo
3. Juliet García Tamayo
4. Daylín Dainery Montero Ramos

Al realizarse las reuniones planificadas, con el objetivo de confirmar que los requisitos poseen los atributos de calidad deseados, se encontraron algunos defectos y errores en el contenido, estos fueron solucionados a medida que se iban desarrollando las diferentes reuniones.

A continuación se muestra una tabla que contiene la lista de errores y acciones recomendadas que se generó luego de realizadas las reuniones correspondientes.

Nº de requisito	Defectos detectados	Acciones recomendadas
5	Realizabilidad	El Sistema no puede Imprimir automáticamente reportes de las métricas calculadas de los proyectos por tipo de características de calidad. Quizás se pueda encontrar otra alternativa. Precisar, en este caso, o eliminar por irreal.
3	Ambigüedad	Especificar el significado de las variables A y B en el cálculo de métricas en la aplicación.
4	Concisión	Separar lo referente a Generar reporte de las métricas por proyecto en dos opciones: Generar reporte general y Generar reporte por especialista.
3	Ambigüedad	Precisar el propósito de las métricas en el cálculo de las

	mismas.

Tabla 3. Lista de errores y acciones recomendadas.

2.3.1.2 Métrica para la calidad de la especificación

Los requisitos identificados fueron comprobados a través de la aplicación de la métrica para la calidad de la especificación con el propósito de evaluar que no exista ambigüedad.

Para determinar la especificidad de los requisitos, sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:

$$Q_1 = \frac{n_{ui}}{n_r}$$

Donde:

Q1: Especificidad de los requisitos para los que todos los revisores tuvieron Interpretaciones idénticas. Cuanto más cerca de uno este el valor de Q1 menor será la ambigüedad de la especificación (eq1sisinf, 2013).

Nr: Total de requisitos definidos.

Nr=51

Nui: Total de requisitos para los que los revisores tuvieron interpretaciones idénticas.

Nui=51

Se obtiene como resultado: Q1= 51/51 = 1

El valor de Q1 resultó ser 1, por lo que se puede concluir que la especificación de los requisitos no posee ambigüedad y presenta un alto grado de especificidad.

2.4 Planificación del proyecto por roles

A continuación se muestran los roles y responsabilidades que intervienen en el desarrollo del sistema.

Rol	Responsabilidad	Nombre
Gerente	Dirigir el equipo, despachar los entregables, negociar con el cliente y tomar las decisiones finales sobre los estándares a cumplir en el proyecto.	Asnier Enrique Góngora Rodríguez Lisandra Díaz Figueredo
Cliente	Participa en las tareas que involucran la	CALISOFT

	lista de reserva del producto.	
Programador	Elaborar el código de los diferentes módulos que se le agregarán a la aplicación, además de los cambios pertinentes según las necesidades del cliente. Mantener una buena comunicación y coordinación con todos los miembros del equipo.	Daylín Dainery Montero Ramos Juliet García Tamayo
Analista	Escribir las historias de usuario y llevar el expediente de proyecto	Daylín Dainery Montero Ramos Juliet García Tamayo
Servicios	Administrar los servicios que se prestan con la aplicación.	Daylín Dainery Montero Ramos Juliet García Tamayo
Diseñador	Diseñar el sistema; los prototipos de interfaces, máximos responsables de realizar el diseño de las metáforas y supervisar el proceso de construcción.	Daylín Dainery Montero Ramos Juliet García Tamayo
Probador	Ayudar al cliente a escribir las pruebas funcionales. Ejecutar las pruebas regularmente, difundir los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Daylín Dainery Montero Ramos Juliet García Tamayo
Arquitecto	Vincularse directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayudar en el diseño de las metáforas.	Daylín Dainery Montero Ramos Juliet García Tamayo

Tabla 4. Roles y responsabilidades.

2.5 Actores del sistema

En la siguiente tabla se muestra la descripción de cada actor del sistema según las actividades que desempeña.

Actor	Descripción
Jefe del laboratorio	Es el encargado de asignar los proyectos que están en pruebas a los especialistas de calidad. Elaborar los reportes de métricas de todos los proyectos que se están revisando, para luego enviárselos a los directivos de la universidad para que ayude en las toma de decisiones.
Especialista de calidad	Encargado de dirigir las pruebas que se le realicen a los proyectos, así como asignar las métricas seleccionadas, recoger los datos y calcular las métricas, además de realizar los reportes por proyectos.
Administrador	Encargado del mantenimiento del sistema así como de gestionar todo el proceso de permisos a los usuarios que acceden al mismo.

Tabla 5. Descripción de actores del sistema.

2.6 Modelo de dominio

El modelo de dominio es utilizado en la investigación para capturar y expresar el entendimiento ganado en el análisis como paso previo al diseño del sistema. En la Figura 2 se muestra el modelo de dominio que es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir**.**

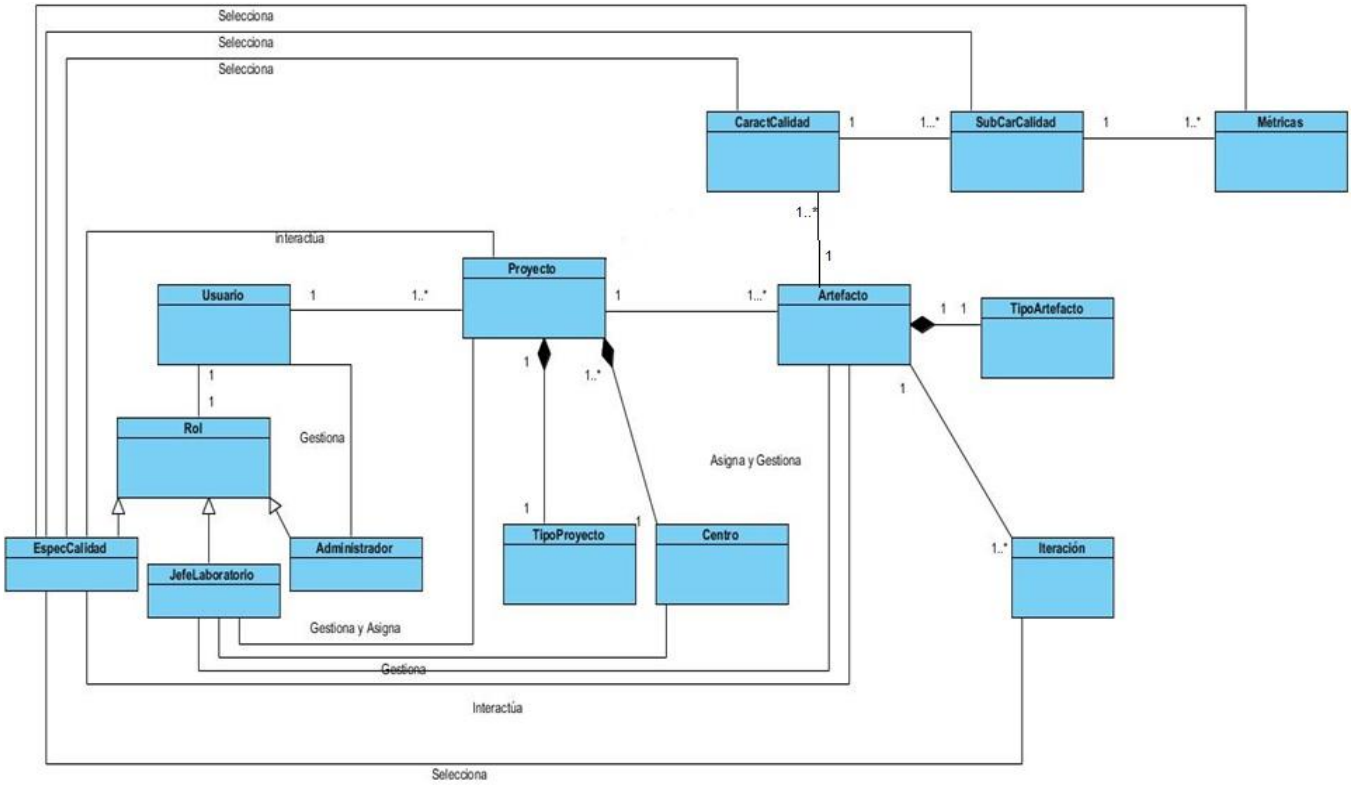


Figura 2. Modelo de dominio.

2.7 Historias de usuarios

Las historias de usuario son la técnica utilizada por la metodología SXP para especificar los requisitos del software, tanto requisitos no funcionales como funcionales. Están escritas en lenguaje del cliente, especifican cada uno de los requisitos del sistema de forma muy breve, sin necesidad de documentaciones extensas y reflejan todas las características del sistema. A continuación se exponen las Historias de Usuarios correspondientes a la herramienta para la gestión de métricas de calidad en las pruebas.

Historia de usuario	
Número: 01	**Usuario:** ddmontero
Nombre historia: Adicionar proyecto	
Prioridad en negocio: Muy alta	**Riesgo en desarrollo:** 4

Puntos estimados: 3	**Iteración asignada:** Primera Iteración
Programador responsable: Daylín	
Descripción: Tiene como objetivo adicionar un nuevo proyecto en el cual se accede al módulo de configuración de la herramienta donde están todos los nomencladores y se selecciona la opción añadir un nuevo proyecto. Se introducen los datos del proyecto. Selecciona la opción de Adicionar Proyecto. Valida los datos. Guarda el Proyecto. Muestra un mensaje de información.	
Observaciones: Para adicionar un proyecto es necesario llenar los campos correctamente.	
Prototipo de interfaces: Ver imagen 6 del anexo 2.	

Historia de usuario	
Número: 06	**Usuario:** ddmontero
Nombre historia: Asignar métricas a un artefacto	
Prioridad en negocio: Muy alta	**Riesgo en desarrollo:** 4
Puntos estimados: 3	**Iteración asignada:** Primera Iteración
Programador responsable: Daylín	
Descripción: Tiene como objetivo introducir los datos de la Asignación. Selecciona la opción de asignar métricas. Valida los datos. Guarda la Asignación. Muestra un mensaje de información.	
Observaciones: Para asignar métricas a un artefacto tiene que haberse adicionado un proyecto.	
Prototipo de interfaces: Ver imagen 5 del anexo 2.	

Se presentan las restantes historias de usuario en el Expediente de proyecto.

2.8 Plan de release

A continuación se muestra un plan de iteraciones para la implementación de las historias de usuario en el sistema, teniendo en cuenta el tiempo que será dedicado por el programador.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total

Iteración 1	En esta iteración se desarrollarán las historias de usuario que tienen prioridad muy alta.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	9 semanas
Iteración 2	En esta iteración se desarrollarán las historias de usuario de alta prioridad, las cuales se irán integrando con las ya realizadas.	14, 15, 16, 17, 18, 19, 20	4 semanas
Iteración 3	En esta iteración se desarrollarán las historias de usuario de prioridad media y se integrarán con las historias de usuario ya implementadas.	21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34	3 semanas
Iteración 4	En esta iteración se desarrollarán las historias de usuario de baja prioridad y se integrarán con todas las anteriores para conformar la herramienta para la gestión de métricas.	35, 36, 37, 38, 39, 40, 41, 42, 43	2 semanas

Tabla 6. Plan de release.

2.9 Base de datos

El sistema de gestión presentará una base de datos donde será almacenada toda la información necesaria para la gestión de las métricas. Este modelo de base de datos es llamado modelo entidad relación, ya que representa la realidad a través de un esquema gráfico empleando la terminología de entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a

resolver con el diagramado, y se distinguen de otros por sus características particulares denominadas atributos. El enlace que rige la unión de las entidades está representado por la relación del modelo.
A continuación se muestra el diagrama entidad-relación:

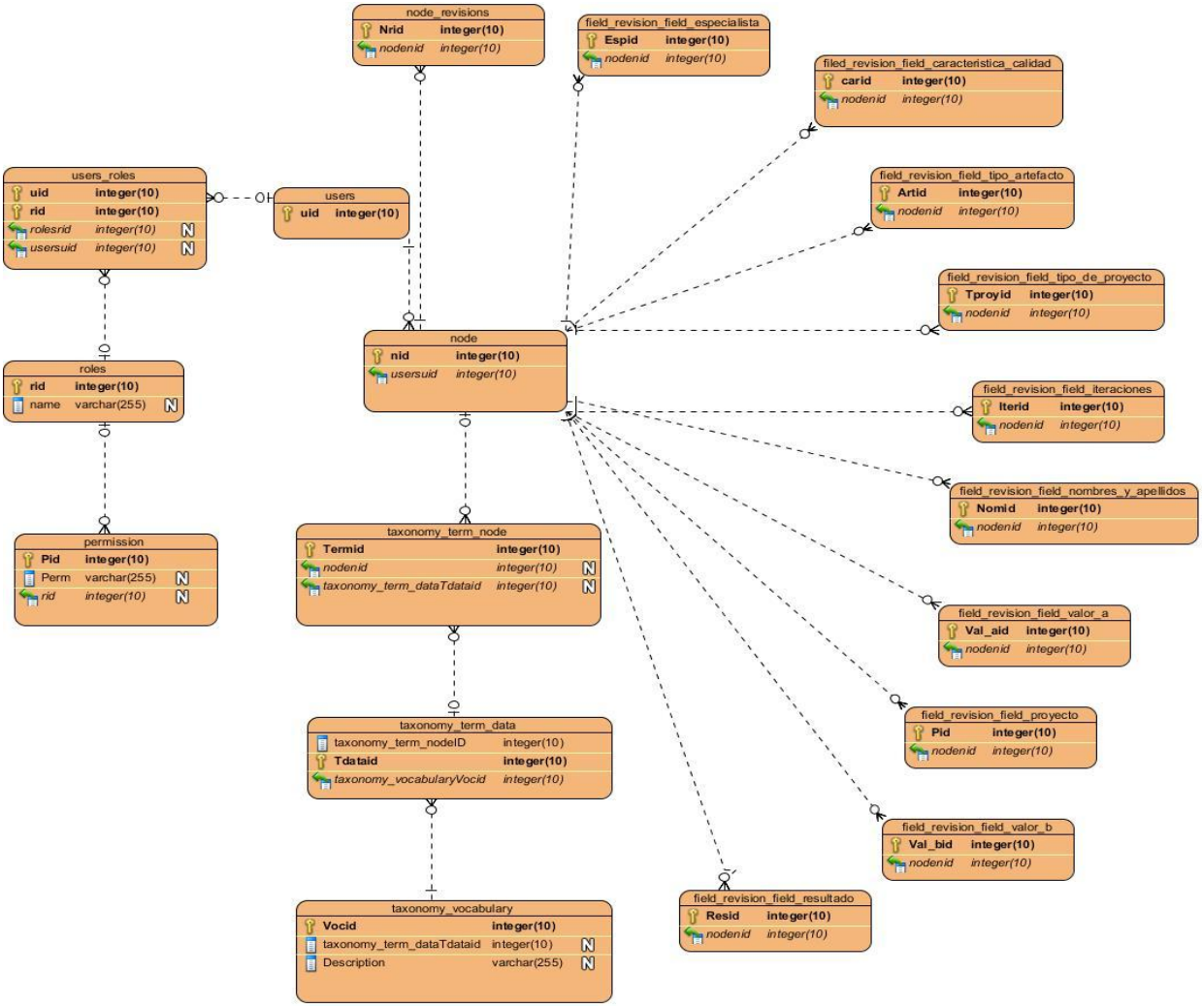


Figura 3. Diagrama Entidad-Relación.

2.10 Arquitectura propuesta

Para la implementación del sistema para la Gestión de métricas de calidad en las pruebas se hace uso del CMS Drupal, el cual se acoge a la arquitectura y patrones que presenta el gestor. Presentando así una arquitectura en 5 capas, como se ilustra en la figura siguiente:

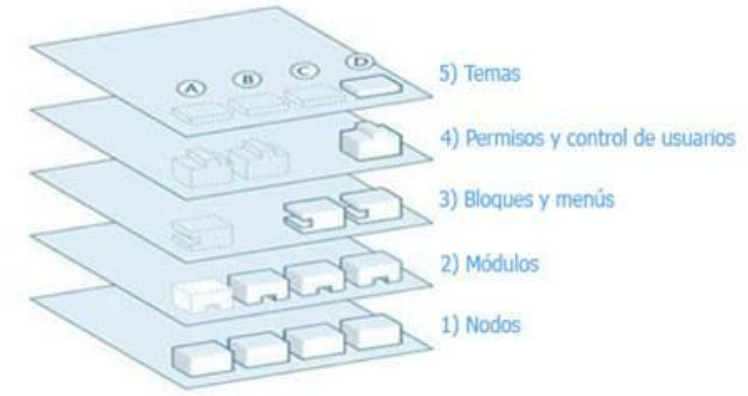


Figura 4. Arquitectura de Drupal (Hechavarría, 2011)

Nodos/*Data*: Estos constituyen la primera capa del subsistema, es donde se almacena la información o los contenidos que en este se maneja.

Módulos/*Modules*: En esta capa se encuentra el módulo implementado y otros complementarios. Estos le otorgarán las funcionalidades al subsistema permitiéndole manipular y operar los nodos que se generen durante la implementación y funcionamiento del mismo.

Bloques y menús/*Blocks & Menus*: Esta tercera capa permite estructurar y organizar la información contenida en el subsistema. Son los bloques y menús los elementos que contienen la mayor parte de la información del subsistema y permiten acceder al usuario a la salida generada y procesada por los módulos que se encuentran en la capa inferior a partir de la información almacenada en los nodos.

Permisos y control de usuarios/*User Permissions*: Esta capa garantiza parte de la seguridad del subsistema debido a que limita las operaciones de lectura, modificación y creación, que se pueden realizar sobre los elementos provenientes de las capas inferiores según los permisos que tengan atribuidos los roles asignados a un determinado usuario; pues el modelo de control de acceso implementado por el CMS es RBAC[6].

Temas/*Template*: Establece la apariencia gráfica o estilo de la información que se le muestra al usuario en el subsistema (Hechavarría, 2011).

[6] Modelo de control de acceso basado en roles.

2.11 Patrones de diseño

Aunque en Drupal no existen clases declaradas explícitamente, presenta características y elementos que lo hacen Orientado a Objetos (OO) como es el caso de los patrones de diseño que este CMS usa.
Entre ellos se encuentran los siguientes patrones GoF:

- **Puente:** La capa de abstracción de bases de datos de Drupal trabaja de forma similar al patrón de diseño puente. Los módulos necesitan ser escritos de una forma independiente a la del sistema de bases de datos que se utiliza y la capa de abstracción proporcionada para ello. Lo que permite que ambas puedan ser modificadas independientemente sin necesidad de alterar por ello la otra además de proporcionar soporte para más sistemas de bases de datos.
- **Cadena de responsabilidades:** El sistema de menú de Drupal de cierta forma sigue este patrón. En cada solicitud que se hace a una página, el menú del sistema determina si existe un módulo que gestione esa solicitud, si el usuario tiene acceso al servicio solicitado y cual función será llamada para satisfacer la petición realizada.
- **Observador:** El patrón Observador es generalizado en Drupal. Cuando una modificación es hecha a un vocabulario en el sistema de taxonomía de Drupal, el gancho taxonomía es llamado en todos los módulos que lo implementan. Mediante la aplicación del gancho, se han registrado como observadores del objeto vocabulario; cualquier cambio a esto puede entonces actuar como es apropiado (Martín, 2010).

  A continuación se muestra un ejemplo de código de la aplicación donde se evidencia el patrón cadena de responsabilidades:

```
function menu_menu() {
$items['admin/structure/menu'] = array(
    'title' => 'Menus',
    'description' => 'Add new menus to your site, edit existing menus, and rename and reorganize
menu links.',
    'page callback' => 'menu_overview_page',
    'access callback' => 'user_access',
    'access arguments' => array('administer menu'),
    'file' => 'menu.admin.inc',
```

```
  );
}
```

Se utilizan además los patrones GRASP, que cumplen con los estándares de implementación y diseño de Drupal:

- **Controlador:** Tiene la responsabilidad de controlar el flujo de eventos del sistema se le otorga a clases específicas, el controlador delega en otras clases las actividades.
- **Experto:** Es el patrón experto en información, define el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.
- **Bajo acoplamiento:** Este patrón se utilizó con la idea de tener las clases lo menos ligadas posible entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- **Creador:** Una clase puede crear un objeto en otra clase solo cuando la contenga, sea agregación o composición de dicha clase.
- **Alta cohesión:** Soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

2.12 Diseño con metáforas

Las metáforas conforman el vocabulario para realizar la descripción del problema, conformando el diseño de solución para determinados momentos del proyecto, generando el Modelo de Diseño que está integrado por un Diagrama de paquetes (Figura 5). Este diagrama muestra los elementos físicos del sistema así como las relaciones existentes entre ellos. Muestra además las dependencias lógicas entre paquetes de software, ya se trate de componentes de código fuente, librerías, entre otros. Normalmente un paquete está pensado como un directorio y están organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes siendo buenos elementos para la gestión.

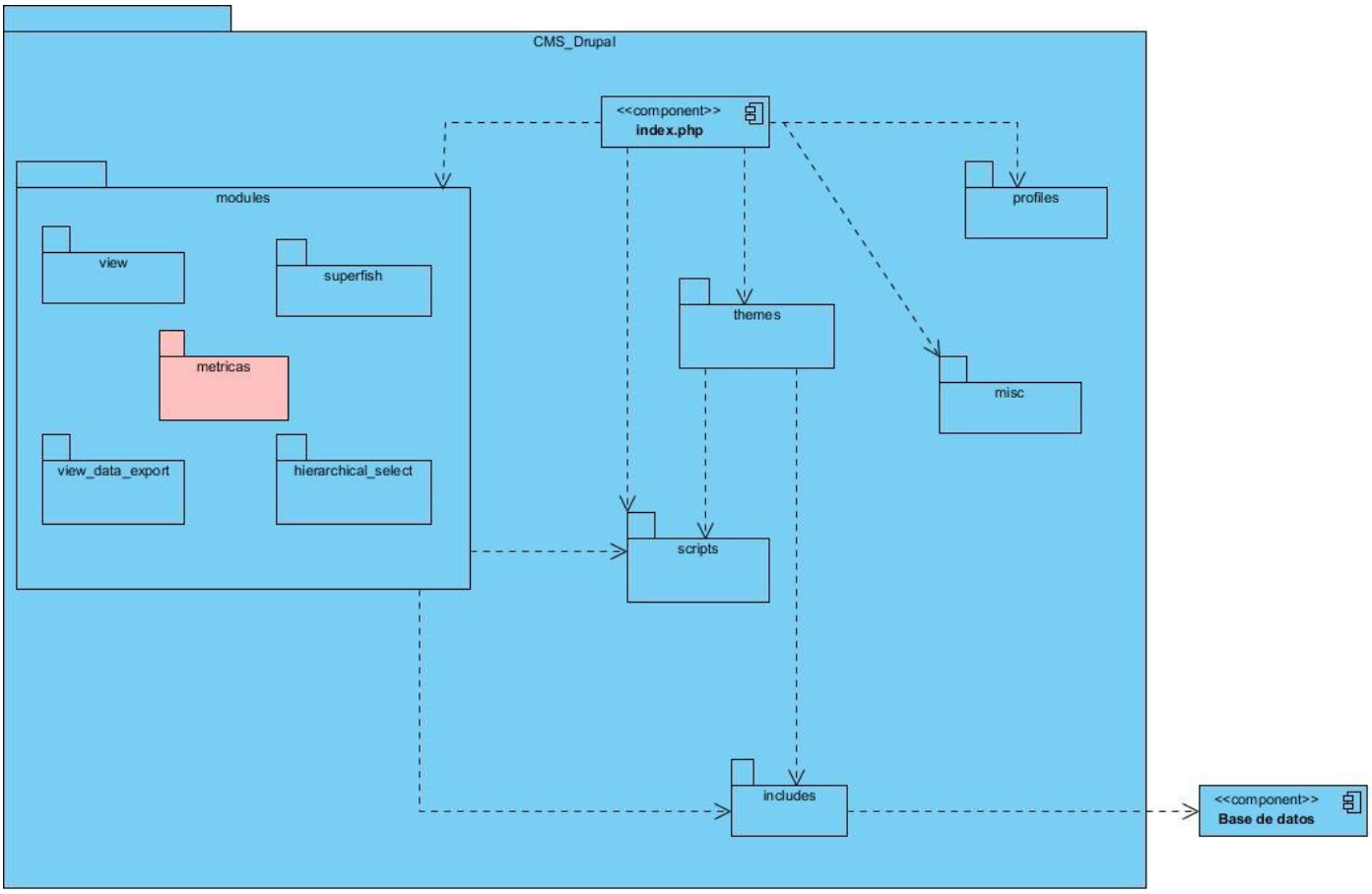


Figura 5. Diagrama de paquetes.

Descripción:

Index: Es el punto principal de entrada para servir peticiones, cada vez que se hace una petición el servidor ejecuta esté componente. A partir de él se invocan los diferentes módulos del CMS.

Include: Es llamado el "corazón de Drupal" ya que contiene la infraestructura del núcleo. En este se encuentran las bibliotecas de las funciones comunes e indispensables para el arranque y funcionamiento del gestor.

Scripts: Contiene secuencias de comandos para el control de la sintaxis, la limpieza de código o ejecutar Drupal desde la línea de comandos manejando casos especiales como el cron.

Themes: Este contiene la plantilla por defecto y temas para Drupal. Permitiendo separar el contenido de la presentación.

Modules: Incluye los módulos por defecto y los adicionales añadidos según la necesidad de los usuarios, los cuales les otorgan diferentes funcionalidades a Drupal.
Misc: Almacena archivos JavaScript y varios íconos e imágenes a disposición de la instalación de Drupal.

Profiles: Contiene los diferentes perfiles de instalación de un sitio permitiendo la instalación de un núcleo y módulos instalarse de forma automática (FABIAN, 2010).

2.13 Diagrama de despliegue

El siguiente diagrama de despliegue muestra la disposición física de los distintos nodos que componen el sistema y el reparto de los componentes sobre dichos nodos. Las conexiones establecidas son asociaciones de comunicación entre los nodos, y se etiquetan con un estereotipo que identifica el protocolo de comunicación.

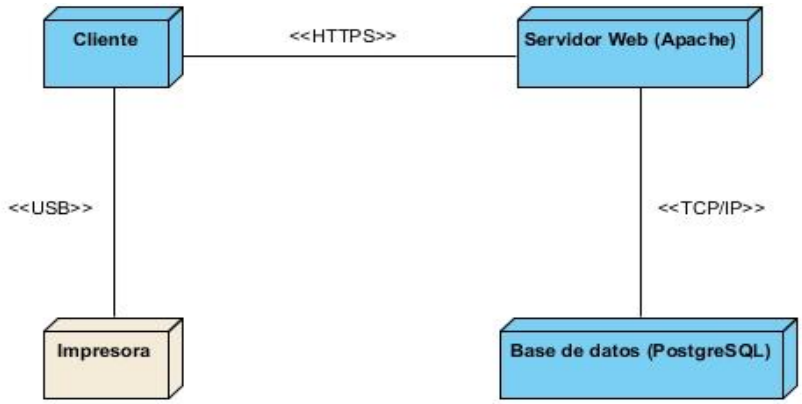


Figura 6. Diagrama de despliegue.

2.14 Conclusiones del capítulo

En este capítulo se abordaron los procesos de diseño e implementación de la herramienta para la gestión de métricas de calidad en las pruebas mediante la metodología ágil SXP.

- La realización del Modelo de Dominio permitió comprender de una mejor forma el negocio relacionado con la gestión de métricas de calidad en las pruebas.
- La elaboración de las historias de usuario, así como la elaboración de las interfaces gráficas, permitieron tener una mejor claridad a la hora de implementar las funcionalidades en el sistema.

- La elaboración del plan de iteraciones para la planificación de la implementación de las historias de usuario, teniendo en cuenta la prioridad de las mismas, propició una mejor organización del trabajo.
- La realización del diseño con metáforas referente al sistema permitió definir la arquitectura del software.

Capítulo 3. Implementación y prueba

En este capítulo se van tratar temas referentes a la implementación de la propuesta y se presentarán los resultados de las verificaciones realizadas durante las pruebas a los proyectos seleccionados, logrando una mayor efectividad en el proceso de pruebas de liberación teniendo en cuenta el cumplimiento de las características de calidad de la norma ISO/IEC 9126-1: 2005 Parte 1: Modelo de Calidad, además se mostrarán tablas con dichos resultados.

3.1 Implementación

En la implementación se comienza con el resultado del análisis y diseño, y se implementa el sistema en términos de componentes, es decir: ficheros de código fuente, scripts, ficheros de códigos binarios y ejecutables. Se generan los artefactos: Plantilla de release y Plantilla de estándar de código que se encuentran en el Expediente de proyecto.

En esta fase se genera todo el código fuente necesario para satisfacer las HU definidas para la solución y se describen todas las tareas realizadas en cada iteración. Al inicio de cada HU, se lleva a cabo una revisión del plan de release y se modifica de ser necesario. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador o grupo de programadores como responsables. Estas tareas, pueden escribirse utilizando un lenguaje técnico y no necesariamente deben ser entendibles para el cliente.

Para la implementación del sitio se determinaron en la fase de planificación cuatro iteraciones de desarrollo. A continuación se muestra una tabla con las tareas de programación correspondiente a la primera iteración.

Iteración	HU	Tareas de programación
	Gestionar usuario	Diseñar la interfaz de la funcionalidad Adicionar usuario. Implementar la funcionalidad Adicionar usuario.

1	Autenticar usuario	Implementar la funcionalidad Autenticar usuario
	Gestionar roles	Diseñar la interfaz de la funcionalidad Adicionar roles. Implementar la funcionalidad Adicionar roles.
	Gestionar métricas	Diseñar la interfaz de la funcionalidad Insertar datos de las métricas. Implementar la funcionalidad Insertar datos de las métricas. Implementar la funcionalidad Calcular métricas. Implementar la funcionalidad Generar reportes de las métricas calculadas por proyectos. Implementar la funcionalidad Imprimir reportes de las métricas calculadas por proyectos. Implementar la funcionalidad Asignar métricas a un artefacto.
	Gestionar proyecto	Diseñar la interfaz de la funcionalidad adicionar proyecto. Implementar la funcionalidad Adicionar proyecto.

	Gestionar tipo de proyecto	Diseñar la interfaz de la funcionalidad Adicionar tipo de proyecto. Implementar la funcionalidad Adicionar tipo de proyecto.
	Gestionar centro	Diseñar la interfaz de la funcionalidad Adicionar centro. Implementar la funcionalidad Adicionar centro.
	Gestionar artefacto	Diseñar la interfaz de la funcionalidad Adicionar artefacto. Implementar la funcionalidad Adicionar artefacto.
	Gestionar tipo de artefacto	Diseñar la interfaz de la funcionalidad Adicionar tipo de artefacto. Implementar la funcionalidad Adicionar tipo de artefacto.

Tabla 7. Tareas de programación de las HU de la primera iteración.

Las restantes tareas de programación se encuentran en el Expediente de proyecto.

3.1.1 Recolección de los datos en los proyectos

Los datos son recogidos mediante la herramienta para la gestión de la métricas propuesto en el capítulo 2, en la Tabla 6 se muestra un ejemplo de la plantilla utilizada para la recogida de los datos. Cada especialista al frente del proyecto que se encuentre en pruebas será el encargado de registrar los datos para posteriormente proceder al cálculo de la métrica de calidad. Los datos son verificados en cuanto a exactitud y a la unidad de medida apropiada especificada en las métricas descritas en el Anexo 3

(Rodríguez, 2011), así como su uniformidad en caso de ser recogidos por más de una persona en las pruebas.

Nombre de la métrica	Variables	Datos
Adecuación funcional	A - Número de problemas detectados en las funcionalidades del sistema. B - Número de especificaciones de requisitos evaluadas.	A = 5 B = 8

Tabla 8. Variables de la métrica de idoneidad.

3.2 Pruebas

Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Las pruebas constituyen el último bastión desde el que se puede evaluar la calidad y, de forma más pragmática, descubrir los errores. La aplicación adecuada de los métodos y de las herramientas, las revisiones técnicas formales efectivas y una sólida gestión y medición, conducen a la calidad, que se confirma durante las pruebas (Pressman, 2005).

3.2.1 Estrategia de prueba

Una estrategia de prueba del software integra las técnicas del diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo, cuándo se deben planificar y realizar esos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes (Pressman, 2005).

3.2.1.1 Niveles de pruebas

Entre los niveles de pruebas se encuentran:

Pruebas de Unidad: Es realizada por el desarrollador y se centra en el módulo. Utilizando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir

errores dentro del ámbito del módulo. La prueba de unidad hace uso intensivo de las técnicas de prueba de caja blanca.

Pruebas de integración: La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. Existen dos formas de integración:

- **Integración descendente**: La prueba de integración descendente es un planteamiento incremental a la construcción de la estructura de programas. Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal (programa principal).
- **Integración ascendente**: La prueba de la integración ascendente, como su nombre indica, empieza la construcción y la prueba con los módulos atómicos (es decir, módulos de los niveles más bajos de la estructura del programa). Dado que los módulos se integran de abajo hacia arriba, el proceso requerido de los módulos subordinados siempre esté disponible y se elimina la necesidad de resguardos.

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

Pruebas de aceptación: Permite que el cliente valide todos los requisitos. Las realiza el usuario final en lugar del responsable del desarrollo del sistema, una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. De hecho, la prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema (Pressman, 2005).

3.2.2 Métodos de pruebas

Existen diferentes métodos de pruebas entre las que se encuentran:

Método de caja blanca: Basada en los requerimientos, en el diseño y en el código fuente.

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

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

Método de caja negra: Es basada en los requerimientos y verifica si el sistema realmente hace lo que debe hacer, es decir, se llevan a cabo sobre la interfaz del software. Es completamente indiferente el comportamiento interno y la estructura del programa.

- **Entradas:** Archivos ejecutables, documentos de requerimientos.
- **Salidas:** Documentos con defectos encontrados. Métricas de la aplicación de casos de pruebas (Pressman, 2005).

3.2.3 Tipos de pruebas

Existen varios tipos de pruebas entre los que se encuentran:

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

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

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

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

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

Prueba función: Es la prueba centrada en validar las funciones que son objeto de prueba como lo que deben ser, ofreciendo los servicios, métodos o casos de usos requeridos. Esta prueba es implementada y ejecutada contra diferentes objetos de pruebas, incluyendo unidades, unidades integradas, aplicaciones y sistemas. Se le aplican a todas o solo un subconjunto de pruebas de función. Puede implicar la re-ejecución de cualquier tipo de prueba. Normalmente, esta prueba se lleva a cabo durante cada iteración, ejecutando otra vez la prueba de la iteración anterior.

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

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

Prueba de carga y stress: Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible). Es la prueba usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema está bajo prueba permanece constante. La variación en carga es simular a la carga de trabajo promedio y con picos que ocurren dentro de tolerancias operacionales normales (Pressman, 2005).

3.2.4 Resultado de las pruebas

Como estrategia de prueba se le aplicó al sistema el nivel de prueba de integración ascendente, donde se realizó el tipo de prueba funcional utilizando el método de prueba de caja negra; luego se realizó el nivel de prueba de sistema, donde se efectuaron las pruebas de seguridad así como las de carga y stress utilizando también el método de caja negra. Este proceso se realizó con el objetivo de que el producto contara con un correcto funcionamiento y a su vez cumpliera con los requisitos indicados por el cliente. Para esto se hizo uso de diferentes herramientas como: JMeter para las pruebas de carga y estrés; y el Acunetix WVS Reporter versión 8.0 para las pruebas de seguridad.

3.2.4.1 Prueba de Carga y Stress

Para realizar las pruebas de carga y estrés se realizó el plan de pruebas en el cual se describen una secuencia de escenarios sobre los cuales se simulará el trabajo de una cantidad determinada de usuarios; para esto se utilizó la herramienta JMeter que permite realizar Pruebas de Rendimiento sobre la aplicación. Para la realización de la misma se le entraron 60 páginas (Hilos), que simulan la cantidad de usuarios que están interactuando con el sistema desde la misma URL a un intervalo de 1 segundo. En este caso son 60 usuarios, pues son la cantidad de usuarios posibles que se pueden conectar de forma concurrente a la aplicación de acuerdo al equipo de trabajo de CALISOFT. Se obtuvo como resultado una Media de 1024 páginas que se cargaron de manera satisfactoria y una Mediana o Tiempo promedio que han tardado en cargarse las páginas de 0.6 segundos. Para 6779 peticiones que se le hicieron al servidor se obtuvo un tiempo mínimo de 0.2 segundos de respuesta y un tiempo máximo de 1.2 segundos. El tiempo total que tardó en responder el 90% de las páginas de manera satisfactoria fue aproximadamente 0.3 segundos y se obtuvo un margen de error de 0.0% demostrando así que el sistema se mantiene funcional aún en situaciones donde concurren gran cantidad de usuarios trabajando sobre ella. El total del

tiempo que demoró en cargarse la cantidad de hilos de esa prueba es menor de 18.5 segundos, lo que demostró una gran capacidad de respuesta y procesamiento del sistema.

En la siguiente tabla se muestran los valores anteriormente descritos:

	#Muestras	Media	Mediana	Mín.	Máx.	Línea de 90%	% Error	Rendimiento
Total	60	1024	06	0.2	1.2	0.3	0.0	18.5

Tabla 9. Resultados de la prueba de carga y stress.

3.2.4.2 Pruebas de seguridad

Para probar la seguridad del sistema se le aplicaron una serie de pruebas usando el software Acunetix WVS Reporter v8.0. En el gráfico 1 se muestran los resultados obtenidos en la tercera iteración de prueba, lo cual muestra que no se detectó ningún riesgo de nivel alto, 5 de nivel medio, 2 de nivel bajo y 23 de tipo informacionales; para un total de 30 alertas encontradas. Luego la aplicación se sometió a un proceso de erradicación de esas vulnerabilidades y se solucionaron las mismas para así concluir que el mecanismo de seguridad del sistema funciona correctamente y garantiza la integridad y disponibilidad de los recursos a gestionar.

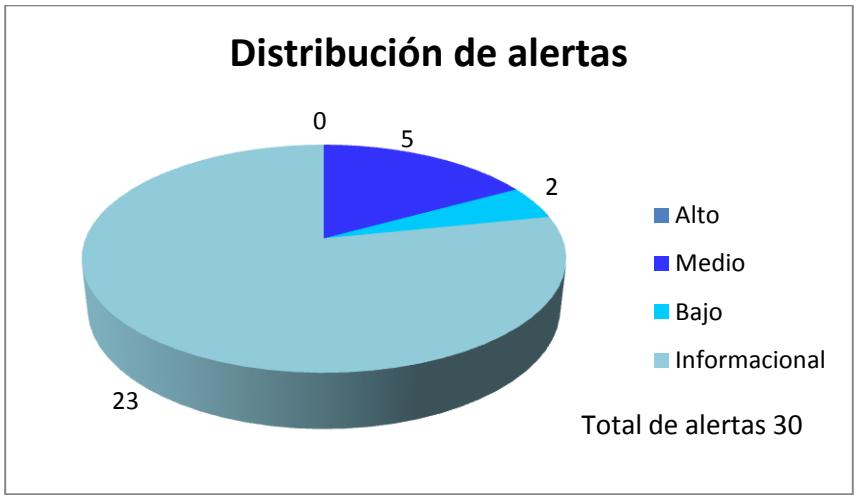


Gráfico 1: Distribución de alertas.

3.2.4.3 Pruebas de funcionalidad

En las diferentes iteraciones de pruebas de funcionalidad se detectaron varias no conformidades de tipo ortografía, validación, funcional e interfaz de usuario, las cuales fueron atendidas y erradicadas, logrando así un correcto funcionamiento de la aplicación. Para esto se hizo uso del método de prueba de caja negra, del cual se utilizaron las técnicas:

- **Partición equivalente:** divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.
- **Análisis de valores límite (AVL):** es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los <<extremos>> de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida (Pressman, 2005).

A continuación se muestra la tabla de las no conformidades encontradas en la realización de las pruebas funcionales de la aplicación.

No conformidades	Iteración 1	Iteración 2	Iteración 3
Ortografía	2	1	0
Validación	5	3	1
Funcional	9	5	2
Interfaz de usuario	2	1	0

Tabla 10. Distribución de no conformidades por iteración de prueba.

3.2.5 Evaluación del sistema aplicando métricas de calidad.

Al sistema se le aplicaron métricas de calidad para verificar la calidad con que cuenta el mismo. Para esto se le aplicaron las características de calidad de funcionalidad y de seguridad. A continuación se muestra una tabla con los resultados obtenidos luego de aplicar las mismas.

Característica de calidad	Subcaracterística de calidad	Métricas	Resultados
Característica de calidad de funcionalidad	Idoneidad	Adecuación funcional	0.7596 (76%)
		Estabilidad en la especificaciones funcionales	0.79844 (80%)
	Funcionalidad	Cumplimiento de estándar de interfaz	0.8888 (89%)
Característica de calidad de seguridad	No tiene	Prevención de la corrupción de los datos	0.92222 (92%)

Tabla 11. Evaluación del sistema aplicando métricas de calidad.

De esta forma se puede concluir que el sistema cuenta con una buena calidad ya que cuenta con una calidad total de 0.8633 (86%) y a mayor proximidad a 1 mejor será el resultado obtenido.

3.5 Validación de la investigación.

Se considera importante que cualquier proceso de investigación se encuentre estrechamente relacionado con el mundo real de aplicación de su sector. Es por eso que se hace el proceso de validación, donde se verifica que el sistema producido cumple con lo planteado en el objetivo general de la investigación, dándole solución al problema a resolver.

La validación es la comprobación de un conjunto de datos para determinar si su valor se halla dentro de unos límites de fiabilidad (Diccionario Enciclopédico Vox 1. Larousse Editorial, 2009). La conexión de la visión teórica procedente de la investigación con la práctica del día a día, se hizo a través de las variables: tiempo y almacenamiento de los datos. A continuación se hace una breve comparación de cómo en CALISOFT se trabaja con estas variables y como sería con la herramienta para la gestión de métricas de calidad en las pruebas de software.

Almacenamiento de los datos.

Actualmente en el centro CALISOFT, el almacenamiento de los datos se hace de forma manual. La recolección de los datos de las métricas y el cálculo de las mismas se realizan en un documento Excel, por consiguiente, los especialistas de calidad para hacer los reportes de las métricas por proyecto tienen que buscar en cada uno de los archivos existentes. Los datos se almacenan en un repositorio, además no se le puede hacer copia de seguridad y si por cualquier motivo a los especialistas de calidad se les olvida subir la información a dicho repositorio, esta información se pierde.

Con la herramienta para la gestión de las métricas de calidad en las pruebas de software, el almacenamiento de los datos se encontraría en un solo lugar, por lo que sería más fácil para los especialistas de calidad realizar los reportes de las métricas por proyecto. Además se le puede hacer copias de seguridad a la base de datos.

Tiempo.

Actualmente en el centro CALISOFT, la recolección de los datos, el cálculo y los reportes de las métricas por proyecto se hacen de forma manual, estos procesos consumen en total un tiempo de 8 horas, lo que conlleva a un retraso en el proceso de pruebas de liberación. Por esta razón en disímiles ocasiones la reunión de cierre, que es donde se dan los resultados de las pruebas realizadas, hay que aplazarla porque no se tiene el dictamen técnico, donde se evidencia el porciento de cumplimiento de las características de calidad.

Con la herramienta para la gestión de las métricas de calidad en las pruebas de software, según la herramienta JMeter, utilizada en la realización de las pruebas de carga y stress, dio como resultado un rendimiento de 18.5 segundos en total.

Se considera que el uso de la herramienta para la gestión de las métricas de calidad en las pruebas de software en el centro CALISOFT es muy conveniente para los especialistas de calidad, pues los procesos de recolección, cálculo y reportes de las métricas por proyecto, serían más rápidos y menos complejos.

3.6 Conclusiones del capítulo

- Luego de generar todo el código fuente necesario para satisfacer las HU definidas para la solución, se describieron todas las tareas realizadas en cada iteración, logrando así obtener una aplicación que cumple con los requisitos del cliente.

- Después de someter a la aplicación a tres iteraciones de pruebas se obtuvieron un total de 31 no conformidades a las cuales se le dio solución permitiendo obtener un sistema con la máxima calidad requerida.
- Luego de analizar los datos obtenidos de las métricas y su aplicación en proyectos seleccionados, se percibió el cumplimiento de cada una de las subcaracterísticas de calidad seleccionadas, para así evaluar los productos y contribuir al proceso de pruebas de liberación en CALISOFT.

Conclusiones generales

- Atendiendo al estado del arte y las condiciones propias del Laboratorio de Pruebas de Software y guiadas por las características de calidad que plantea el modelo ISO/IEC 9126 se establecieron un conjunto de métricas de calidad para el proceso de pruebas.
- Las herramientas, tecnologías y metodología utilizadas satisfacen las necesidades del sistema a desarrollar teniendo en cuenta los requisitos funcionales y no funcionales.
- Mediante la implementación de la herramienta informática se logró elevar la eficiencia en el proceso de liberación de los productos, disminuyendo los tiempos de respuesta a las solicitudes de los clientes.
- Las pruebas realizadas a la herramienta de gestión de métricas de calidad permitieron validar el correcto funcionamiento del producto respecto a los requisitos del cliente.

Recomendaciones

- Agregar a la herramienta para la gestión de las métricas una nueva funcionalidad que permita adicionar nuevas métricas de calidad definidas para luego calcularlas y tener más mediciones en el proceso de prueba.

Referencias bibliográficas

Gómez, David Alejandro. 2009. ISO 9126. [En línea] 25 de noviembre de 2009. http://alejandrogomeziso.blogspot.com.

Moreno García, María N., García Peñalvo, Francisco J. y Polo Martín, María José. 2009. *Medición de la calidad del software en el ámbito de la especificación de requisitos.* s.l. : Universidad de Salamanca.Departamento de Informática y Automática, 2009.

Planning Extreme Programming. Addison Wesley. **Kent Beck, M.F. 2000..** 2000.

Aguilar, Vicente y Suau, Pablo. 2000. MySQL vs. PostgreSQL. [En línea] 18 de agosto de 2000. http://www.bisente.com/documentos/mysql-postgres.html.

Amaro Calderón, Sarah Dámaris y Valverde Rebaza , Jorge Carlos. 2007. *Metodologías Ágiles.* Trujillo – Perú : s.n., 2007.

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

Definición de una metodología ágil de ingeniería de requerimientos para empresas emergentes de desarrollo de software del sur-occidente colombiano. **Merchán, Luis, Urrea, Alba y Rebollar, Rubén. 2008.** No. 1, Colombia : s.n., 2008, Vol. Vol. 6.

Diccionario Enciclopédico Vox 1. Larousse Editorial, S.L. 2009. [En línea] 2009. http://es.thefreedictionary.com.

eq1sisinf. 2013. Métricas de la calidad de la especificación. [En línea] 2013. eq1sisinf - 4.1.3 Métricas de la calidad de la especificación.htm.

García Sánchez, Ana María. 2010. *Evaluación de métricas de calidad del software sobre un programa Java.* Madrid : s.n., 2010.

Hechavarría, Jorge Jesús Pérez. 2011. *Paquete de servicios de valor agregado para la red social universitaria de la Universidad de las Ciencias Informáticas.* 2011.

Henst, Christian Van Der. 2001. Maestros del web. [En línea] 23 de mayo de 2001. http://www.maestrosdelweb.com/editorial/phpintro/.

IBM. 2008. IBM Rational Quality Manager. [En línea] Septiembre de 2008. ftp://ftp.support.lotus.com.

Joomla, Sitio oficial de. enero. Sistema de Administración de Contenido. [En línea] 2013 de enero. http://www.joomlaspanish.org/.

La norma ISO/IEC 25000 y el proyecto KEMIS para su automatización con software libre. **Marcos, José , y otros. 2008.** No. 2, España : REICIS, 2008, Vol. Vol.4.

Lovelle, Juan Manuel Cueva. 1999. *Calidad del Software.* España : s.n., 1999.

MADEJA. 2013. *Técnicas de validación de requisitos .* ANDALUCÍA : s.n., 2013.

Martín, Liber Matos. 2010. POO y patrones de diseño en DRUPAL. [En línea] septiembre de 2010. http://jitcode.blogspot.com/2010/06/poo-y-patrones-de-diseno-en-drupal.html.

Martinez, Rafael. 2010. PostgreSQL-es. [En línea] 2 de octubre de 2010. http://www.postgresql.org.es/sobre_postgresql.

Master de Ingeniería de Software-A Distancia. 2012. *VALIDACION DE REQUISITOS.* 2012.

Mendoza, Gonzalo Mena. 2006. Métricas Internas de la Calidad del Producto de Software. [En línea] marzo de 2006. http://mena.com.mx/gonzalo/maestria/calidad/presenta/iso_9126-3/.

Monterrey, Tecnológico de. *Aplicación de la Calidad del Software en el Proceso de Desarrollo .*

Mulén Mustelier, Evelyn y López Salar, Yoan. 2010. *Procedimiento para la evaluación de la usabilidad de los productos software desarrollados en el CESIM.* La Habana : s.n., 2010.

Olsina, Dr. Luis. 2003. Métricas e Indicadores: Dos Conceptos Claves para Medición y Evaluación. [En línea] 2003. http://gidis.ing.unlpam.edu.ar.
Oracle Corporation. 2012. NetBeans IDE. [En línea] 2012. http://netbeans.org/features/index.html.
Paradigm, Visual. 2012. software.com.ar. [En línea] 2012. http://www.software.com.ar/visual-paradigm-para-uml.html.
Pressman, Roger S. 2005. *Ingenieria de Software. Un enfoque práctico.* Ciudad de La Habana : Editorial Félix Varela, 2005.
Rodríguez, Ing. Asnier Enrique Góngora. 2011. *Catálogo automatizado de métricas de calidad para evaluar los productos en las pruebas.* La Habana : s.n., 2011.
Romero, Gladys Marsi Peñalver, Puente, Sergio Jesus García De La y Abad, Abel Meneses. 2010. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* La Habana : s.n., 2010.
Software Quality Systems S. A. 2013. SQS. [En línea] 2013. http://www.sqs.es.
SQS. [En línea] http://www.sqs.es.

Bibliografía

Gómez, David Alejandro. 2009. ISO 9126. [En línea] 25 de noviembre de 2009. http://alejandrogomeziso.blogspot.com.

Moreno García, María N., García Peñalvo, Francisco J. y Polo Martín, María José. 2009. *Medición de la calidad del software en el ámbito de la especificación de requisitos.* s.l. : Universidad de Salamanca.Departamento de Informática y Automática, 2009.

Planning Extreme Programming. Addison Wesley. **Kent Beck, M.F. 2000..** 2000.

Aguilar, Vicente y Suau, Pablo. 2000. MySQL vs. PostgreSQL. [En línea] 18 de agosto de 2000. http://www.bisente.com/documentos/mysql-postgres.html.

Amaro Calderón, Sarah Dámaris y Valverde Rebaza , Jorge Carlos. 2007. *Metodologías Ágiles.* Trujillo – Perú : s.n., 2007.

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

Definición de una metodología ágil de ingeniería de requerimientos para empresas emergentes de desarrollo de software del sur-occidente colombiano. **Merchán, Luis, Urrea, Alba y Rebollar, Rubén. 2008.** No. 1, Colombia : s.n., 2008, Vol. Vol. 6.

Diccionario Enciclopédico Vox 1. Larousse Editorial, S.L. 2009. [En línea] 2009. http://es.thefreedictionary.com.

eq1sisinf. 2013. Métricas de la calidad de la especificación. [En línea] 2013. eq1sisinf - 4.1.3 Métricas de la calidad de la especificación.htm.

García Sánchez, Ana María. 2010. *Evaluación de métricas de calidad del software sobre un programa Java.* Madrid : s.n., 2010.

Hechavarría, Jorge Jesús Pérez. 2011. *Paquete de servicios de valor agregado para la red social universitaria de la Universidad de las Ciencias Informáticas.* 2011.

Henst, Christian Van Der. 2001. Maestros del web. [En línea] 23 de mayo de 2001. http://www.maestrosdelweb.com/editorial/phpintro/.

IBM. 2008. IBM Rational Quality Manager. [En línea] Septiembre de 2008. ftp://ftp.support.lotus.com.

Joomla, Sitio oficial de. enero. Sistema de Administración de Contenido. [En línea] 2013 de enero. http://www.joomlaspanish.org/.

La norma ISO/IEC 25000 y el proyecto KEMIS para su automatización con software libre. **Marcos, José , y otros. 2008.** No. 2, España : REICIS, 2008, Vol. Vol.4.

Lovelle, Juan Manuel Cueva. 1999. *Calidad del Software.* España : s.n., 1999.

MADEJA. 2013. *Técnicas de validación de requisitos .* ANDALUCÍA : s.n., 2013.

Martín, Liber Matos. 2010. POO y patrones de diseño en DRUPAL. [En línea] septiembre de 2010. http://jitcode.blogspot.com/2010/06/poo-y-patrones-de-diseno-en-drupal.html.

Martinez, Rafael. 2010. PostgreSQL-es. [En línea] 2 de octubre de 2010. http://www.postgresql.org.es/sobre_postgresql.

Master de Ingeniería de Software-A Distancia. 2012. *VALIDACION DE REQUISITOS.* 2012.

Mendoza, Gonzalo Mena. 2006. Métricas Internas de la Calidad del Producto de Software. [En línea] marzo de 2006. http://mena.com.mx/gonzalo/maestria/calidad/presenta/iso_9126-3/.

Monterrey, Tecnológico de. *Aplicación de la Calidad del Software en el Proceso de Desarrollo .*

Mulén Mustelier, Evelyn y López Salar, Yoan. 2010. *Procedimiento para la evaluación de la usabilidad de los productos software desarrollados en el CESIM.* La Habana : s.n., 2010.

Olsina, Dr. Luis. 2003. Métricas e Indicadores: Dos Conceptos Claves para Medición y Evaluación. [En línea] 2003. http://gidis.ing.unlpam.edu.ar.

Oracle Corporation. 2012. NetBeans IDE. [En línea] 2012. http://netbeans.org/features/index.html.

Paradigm, Visual. 2012. software.com.ar. [En línea] 2012. http://www.software.com.ar/visual-paradigm-para-uml.html.

Pressman, Roger S. 2005. *Ingenieria de Software. Un enfoque práctico.* Ciudad de La Habana : Editorial Félix Varela, 2005.

Rodríguez, Ing. Asnier Enrique Góngora. 2011. *Catálogo automatizado de métricas de calidad para evaluar los productos en las pruebas.* La Habana : s.n., 2011.

Romero, Gladys Marsi Peñalver, Puente, Sergio Jesus García De La y Abad, Abel Meneses. 2010. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* La Habana : s.n., 2010.

Software Quality Systems S. A. 2013. SQS. [En línea] 2013. http://www.sqs.es.

SQS. [En línea] http://www.sqs.es.

Ana María García Sánchez. Curso 2009-2010. *EVALUACIÓN DE MÉTRICAS DE CALIDAD DEL SOFTWARE SOBRE UN PROGRAMA JAVA.* Facultad de Informática. Universidad Complutense de Madrid.

Patricio Letelier. *METODOLOGÍAS ÁGILES Y XP.* Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia.

Patricio Letelier y Mª Carmen Penadés. *MÉTODOLOGÍAS ÁGILES PARA EL DESARROLLO DE SOFTWARE: EXTREME PROGRAMMING (XP).* Universidad Politécnica de Valencia Camino de Vera s/n, 46022 Valencia {letelier, mpenades}@dsic.upv.es.

Querétaro, marzo de 2006. *ISO 9126-3: MÉTRICAS INTERNAS DE LA CALIDAD DEL PRODUCTO DE SOFTWARE.*

María Antonieta Abud Figueroa. *CALIDAD EN LA INDUSTRIA DEL SOFTWARE. LA NORMA ISO-9126.* La Norma ISO/IEC 9126. *APLICACIÓN DE LA CALIDAD DEL SOFTWARE EN EL PROCESO DE DESARROLLO.*

Peñalver, G. Meneses, A. García, S. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Moreno García, María N., García Peñalvo, Francisco J. y Polo Martín, María José. 2009. *Medición de la calidad del software en el ámbito de la especificación de requisitos.* s.l. : Universidad de Salamanca.Departamento de Informática y Automática, 2009.

Raúl Monferrer Agut. *ESPECIFICACIÓN DE REQUISITOS SOFTWARE SEGÚN EL ESTÁNDAR DE IEEE 830. E78.* INGENIERÍA DEL SOFTWARE 5º Curso de Ingeniería Informática 2000-2000.

Anexos

Anexo 1: Prototipos de interfaz

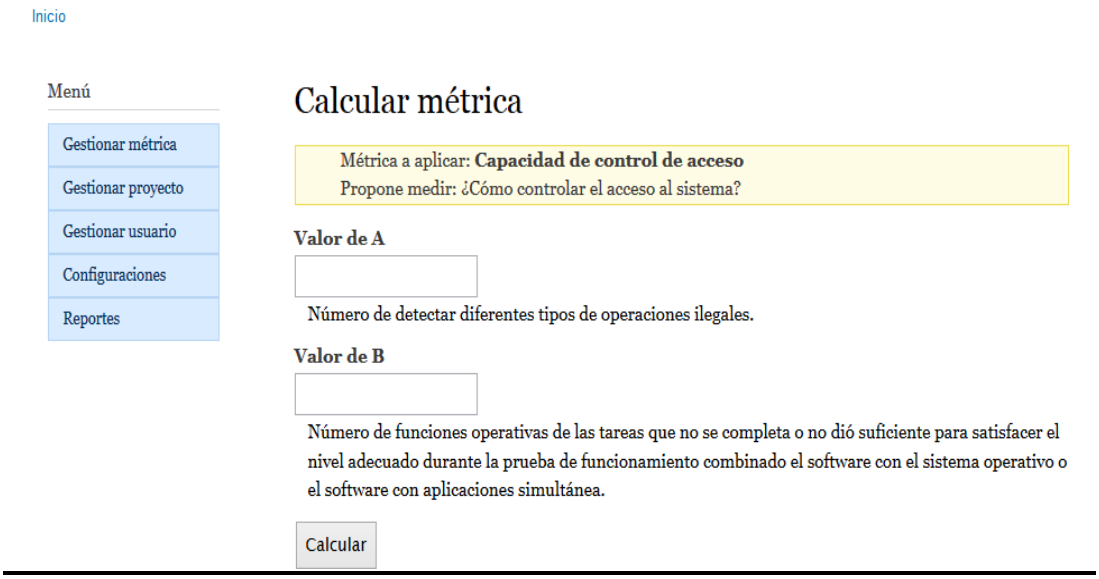


Imagen 1. Calcular métrica.

Imagen 2. Gestionar usuario.

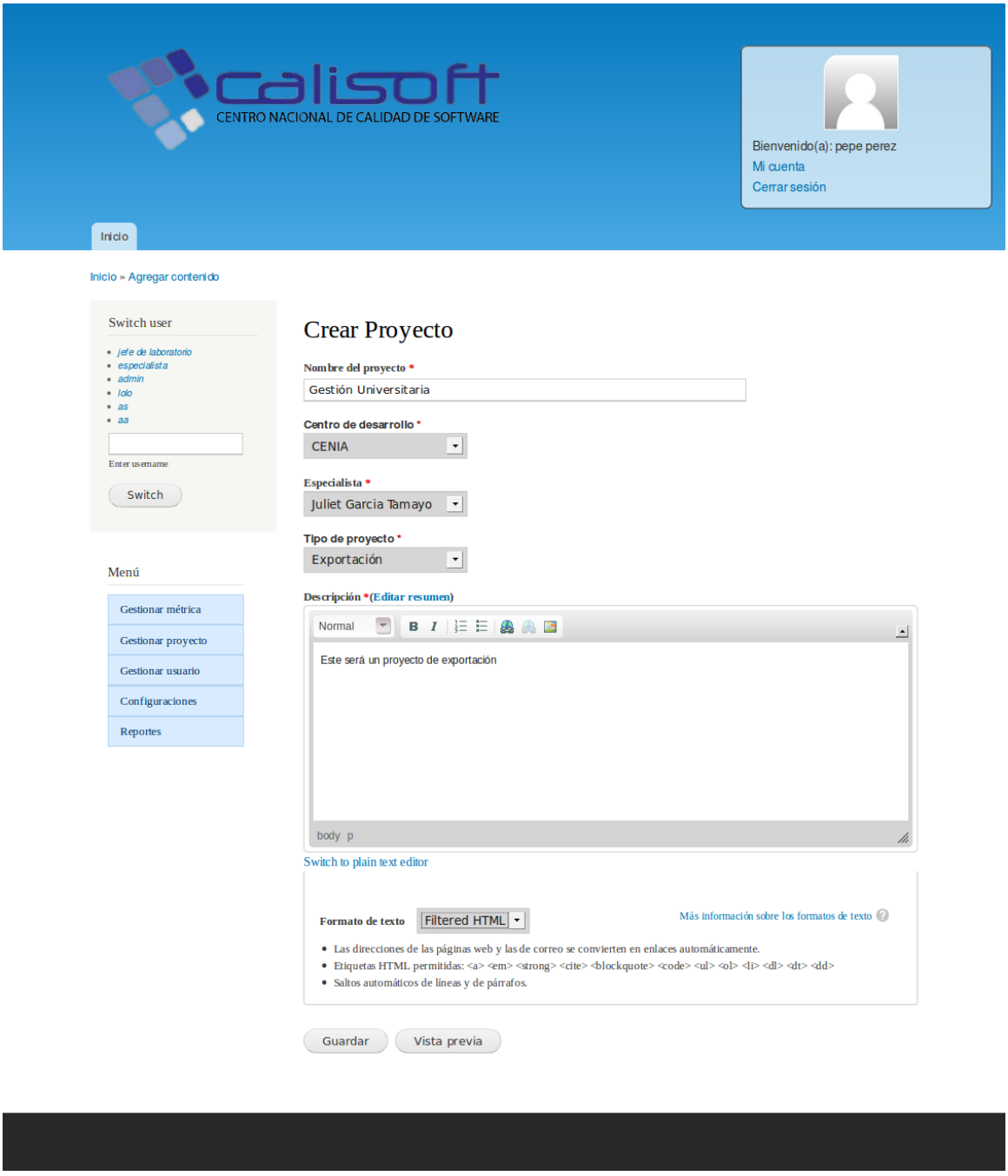


Imagen 3. Adicionar proyecto.

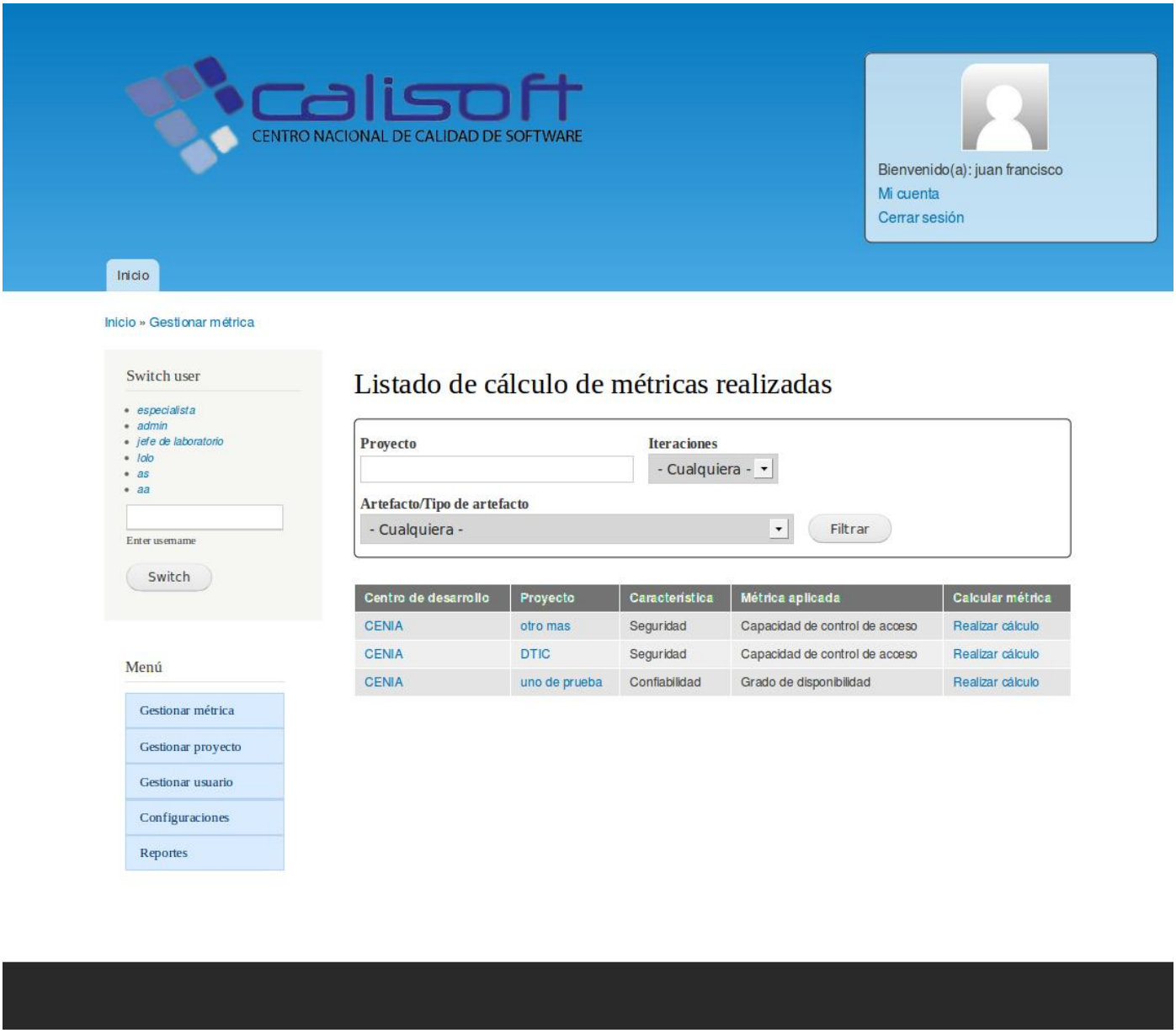


Imagen 4. Listado de métricas calculadas.

Anexo 2: Métricas a utilizar

1. **Madurez de las pruebas** (Propone medir: ¿Está bien probado el producto?).

2. **Densidad de fallos totales contra casos de** prueba (Propone medir: ¿Cuántos fallos totales fueron detectados durante un período de pruebas definido?).
3. **Evitar el fracaso** (Propone medir: ¿Cómo los patrones de muchas fallas fueron puestos bajo control para evitar fallos críticos y graves?).
4. **Evitación de operaciones incorrectas** (Propone medir: ¿Cuántas funciones están implementadas con capacidad de evitación de operaciones incorrectas?).
5. **Grado de disponibilidad** (Propone medir: ¿Cuán disponible está el sistema para su uso durante un período de tiempo especificado?).
6. **Restaurabilidad** (Propone medir: ¿Cuán capaz es el producto de auto restaurarse luego de un evento anormal o una solicitud?).
7. **Tiempo de respuesta** (Propone medir: ¿Cuál es el tiempo necesario para completar una tarea específica? ¿Cuánto tiempo se tarda antes de que la respuesta del sistema a una operación especificada?).
8. **Cumplimiento de la eficiencia** (Propone medir: ¿Cómo es compatible la eficacia del producto a las normas, los estándares y regulaciones?).
9. **Rendimiento** (Propone medir: ¿Cómo muchas tareas pueden realizarse con éxito en un período determinado de tiempo?).
10. **Capacidad de control de acceso** (Propone medir: ¿Cómo controlar el acceso al sistema?).
11. **Prevención de la corrupción de los datos** (Propone medir: ¿Cuál es la frecuencia de eventos de corrupción de los datos?).
12. **Exactitud esperada** (Propone medir: ¿Existen diferencias entre los resultados actuales y los razonablemente esperados?).

13. **Precisión** (Propone medir: ¿Con qué frecuencia los usuarios finales encuentran resultados con la precisión adecuada?).
14. **Cumplimiento de estándar de interfaz** (Propone medir: ¿Cómo se cumplen las interfaces con los reglamentos aplicables, las normas y regulaciones?).
15. **Cumplimiento funcional** (Propone medir: ¿Cómo es compatible con la funcionalidad del producto a las normas aplicables, los estándares y regulaciones?).
16. **Adecuación funcional** (Propone medir: ¿Cuán adecuada es la función evaluada?).
17. **Completitud de la implementación funcional** (Propone medir: ¿Cuán completa ha sido la implementación y su conformidad con la especificación de requisitos?).
18. **Capacidad de adaptación al entorno del software** (Propone medir: ¿Puede el usuario o desarrollador de software adaptarse fácilmente al entorno?).
19. **Facilidad de instalación** (Propone medir: ¿El usuario puede fácilmente instalar en el entorno de funcionamiento?).
20. **Adaptabilidad de la apariencia de la interfaz** (Propone medir: ¿Qué proporción de los elementos de la interfaz puede ser, por su apariencia, adaptado por el usuario para la satisfacción del mismo?).
21. **Accesibilidad a demos** (Propone medir: ¿A qué proporción de demos/tutoriales pueden acceder los usuarios?).
22. **Comprensibilidad de la función** (Propone medir: ¿Qué proporción de las funciones de producto el usuario podrá entender correctamente?).
23. **Eficiencia de la documentación del usuario y/o sistemas de ayuda en el** uso (Propone medir: ¿Qué proporción de las funciones pueden ser utilizados correctamente después de leer la documentación o el uso de sistemas de ayuda?). (Rodríguez, 2011)

Anexo 3: **Métricas por tipo de pruebas**

Tipo de pruebas	Nombre de la métrica
Funcionales	Métrica # 1: Adecuación funcional Métrica # 2: Completitud de la implementación funcional Métrica # 3: Exactitud esperada Métrica # 4: Precisión Métrica # 5: Cumplimiento funcional Métrica # 6: Cumplimiento de estándar de interfaz Métrica # 7: Densidad de fallos totales contra casos de prueba Métrica # 8: Madurez de las pruebas
Seguridad	Métrica # 9: Prevención de la corrupción de los datos Métrica # 10: Capacidad de control de acceso
Recuperación y tolerancia a fallas	Métrica # 11: Evitación de operaciones incorrectas Métrica # 12: Evitar el fracaso Métrica # 13: Grado de disponibilidad Métrica # 14: Restaurabilidad
Usabilidad	Métrica # 15: Accesibilidad a demos Métrica # 16: Comprensibilidad de la función Métrica # 17: Adaptabilidad de la apariencia de la interfaz Métrica # 18: Eficiencia de la documentación del usuario y / o sistemas de ayuda en el uso

Carga y Estrés	Métrica # 19: Tiempo de respuesta Métrica # 20: Rendimiento Métrica # 21: Cumplimiento de la eficiencia
Instalación y Configuración	Métrica # 22: Capacidad de adaptación al entorno del software Métrica # 23: Facilidad de instalación

Tabla 11. Métricas por tipo de pruebas