UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

SofQualityAssess

Sistema informático de ayuda a la decisión para la evaluación de la calidad de productos de software

Trabajo de Diploma para optar por el título de:

Ingeniero en Ciencias Informáticas

Autores: David Pino González

Tony Castillo Martin

Tutor: MSc. Yamilis Fernández Pérez, Ing.

Ing. Luis Manuel Valera Pérez

La Habana, julio de 2016

"Año 58 de la Revolución"

Los que no puedan mantener el ritmo de la revolución tecnológica, se encontrarán con que ellos mismos se han vuelto obsoletos.

Katherine Neville

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores del presente trabajo y se reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firman la presente a los ___ días del mes de _____ del año _____.

_____ _____

David Pino González Tony Castillo Martin

_____ _____

MSc. Yamilis Fernández Pérez, Ing.
Tutor

Ing. Luis Manuel Valera Pérez
Tutor

DATOS DE CONTACTO

Generales del tutor

Nombre y apellidos: Yamilis Fernández Pérez.
Especialidad: Ingeniería en Sistema Automatizado de Dirección
Síntesis del (los) tutor(es):
Graduada de Ingeniera en Sistema Automatizado de Dirección (SAD), del ISPJAE en el año 1992. Máster en Ciencias en Informática Aplicada en la especialidad de Ingeniería de Software en el ISPJAE en el año 1995. Es profesora auxiliar, ha impartido pregrado y posgrado de Ingeniería de Software, Bases de Datos y Programación. Desde el 1992 hasta 1999 profesor de la Universidad de Matanzas, desde 1999 hasta 2002 profesor del ISPJAE. Desde el 2002 hasta la actualidad profesor de la UCI. Fue Jefe de Departamento de la Disciplina Ingeniería y gestión de software desde el 2002 y hasta el 2008 a nivel central. Desde el 2008 hasta el 2012 fue directora de formación posgraduada de la UCI. Actualmente desarrolla el doctorado en Informática en colaboración ISPJAE, UCLV, UCI y Universidad de Granada.
Correo electrónico: yamilisf@uci.cu
Teléfono de contacto:

Generales del tutor

Nombre y apellidos: Luis Manuel Valera Pérez
Especialidad: Ingeniería en Ciencias Informáticas
Síntesis del (los) tutor(es):
Graduado en el curso 2014-2015 como Ingeniero en Ciencias Informáticas. Profesor de Sistemas de Base de Datos de la Facultad 5. Recientemente obtuvo la categoría de Profesor Instructor.
Correo electrónico: luismvalera@uci.cu
Teléfono de contacto: 837-2219

AGRADECIMIENTOS

David:

Primeramente, quiero agradecerle a mi difunta abuela Nieves que, aunque no se encuentra entre nosotros sé que me sigue apoyando como lo hizo en vida.

Quiero agradecerles a mis padres, a mis hermanas y hermano por, cuidarme, educarme y estar siempre presentes cuando más los necesito.

Quiero expresar mi agradecimiento a mis familiares de aquí de La Habana por ayudarme durante mi estancia, y por brindarme un espacio en sus corazones.

Le agradezco a mi compañero de tesis Tony y a mis tutores Yamilis y Luis Manuel por su apoyo y dedicación en la tesis.

También quiero agradecer a mi novia, mis amigos más allegados a los cuales considero como hermano. También a mis compañeros de apartamento, de grupo, y del equipo de Básquet por compartir momentos inolvidables.

Le agradezco a los profesores que durante la carrera influyeron de una forma u otra en mi preparación como profesional.

Tony:

A mi abuela Tita, gracias por todo su amor y por siempre estar ahí para mí.

A mi mamá por amarme tanto, por dedicarme su vida entera, por tener el corazón más grande y hermoso del mundo, por ser mi ejemplo de fuerza, de trabajo, de sacrificio, de amor, hoy soy una mejor persona gracias a ti, a ti que te debo todo lo que soy, mamá, no me alcanzan las palabras, para mí, eres lo más grande del mundo.

A mi hermana por tanto amor, por siempre hacerme feliz, por compartir tantos momentos difíciles y buenos juntos, eres parte de mí ñiñi.

A mi novia, la mujer más linda del mundo, gracias por estar conmigo todo este tiempo, gracias por ser mi luz en la oscuridad, siempre que he estado triste o cansado tú me curas con esa sonrisa mágica que me alegra el alma, gracias por quererme tanto, por ser tan atenta conmigo, y a su familia como olvidarla si desde que llegue un buen día me acogieron como uno más de ellos, de lo que estoy totalmente agradecido.

A mis tíos, mis primos y en fin a mi familia en general.

A mi tutora Yamilis, a ella le debo gran parte de esta investigación, gracias a su insistencia y consejos, hoy estoy aquí defendiendo mi tesis.

A mi tutor Luis Manuel, que no fue solo mi tutor, sino hermano mayor, que estuvo apoyándome todo este tiempo, dando consejos y escuchándome, gracias por todo.

A mis amigos que durante estos años han sido mi familia y que siempre han estado presentes para mí.

A los profesores que ayudaron a nuestra formación personal.

A mi compañero de tesis David, por ser el amigo que es, a él y a su familia por su cariño.

DEDICATORIA

Tony:

A mi mamá y a mi abuela, quienes me han forjado, guiado y apoyado durante todos los años de vida que tengo. Por ser los principales impulsores a que estudiara y pudiera ser hoy una profesional, a mi hermana y mi novia por estar conmigo siempre, y a toda mi familia en general.

David:

A mi abuela Nieves, a mi mamá, a mi hermana Diyankas y a mi tia Nivia, por darme su amor y comprensión, por educarme y ayudarme en los momentos difíciles. Por ser las mujeres más valientes y fuertes que me han guiado durante el transcurso de mi vida.

RESUMEN

La calidad del software es una preocupación a la que se dedican muchos esfuerzos. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios. La presente investigación tiene como precedente la necesidad de CALISOFT de realizar las evaluaciones de calidad de los productos de software usando diferentes técnicas de agregación de la información a través de los métodos de toma de decisión multicriterio (MCDM) y comparar los diferentes rankings obtenidos por los métodos aplicados. Para ello es esencial el uso de un sistema automatizado que guie el proceso de agregación de la información. En la investigación se obtiene una aplicación informática que contiene los métodos de decisión multicriterio como AHP, ANP, TOPSIS, Suma Ponderada y un método nuevo denominado Versus. El sistema se desarrolló a partir de tecnologías libres, con XP como metodología de desarrollo y varias herramientas necesarias para la validación y correcto funcionamiento del mismo.

Palabras clave: Evaluación de la calidad de software, mapas cognitivos difusos, método de decisión multicriterio, toma de decisión

Índice de Contenido

Introducción

La calidad del software es una preocupación a la que se dedican muchos esfuerzos. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios. (1)

La calidad del software es el grado en que el producto de software satisface las necesidades expresadas o implícitas, cuando es usado bajo condiciones determinadas" (NC ISO/IEC 25000: 2011). Por lo que, la calidad la definen un conjunto de cualidades que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad (2).

La importancia del proceso de evaluación de la calidad del software radica en el hecho de que, a través de una serie de actividades, procedimientos y procesos orientados a lograr la característica importante del producto y del servicio de satisfacer los requisitos del cliente. Además, puede expresar la comparación entre la calidad real y la calidad propuesta o esperada. Este proceso se puede definir como complejo, debido a que maneja mucha información de diferentes orígenes y especialistas por lo que es esencial automatizar el proceso para poder tomar decisiones efectivas y menos costosas.

La Universidad de las Ciencias Informáticas (UCI) a partir del año 2004, comienza a tener una presencia productiva en la industria del software con el desarrollo de proyectos informáticos, fundamentalmente relacionados con la informatización de la sociedad y la exportación; por tal motivo, más del 70% de los estudiantes se encuentran incorporados a proyectos productivos e investigativos de software. Dentro de la infraestructura productiva de la universidad está la necesidad de medir la calidad del software, es por eso que surge el Centro Nacional de Calidad de Software (CALISOFT), adscrito al Ministerio de Comunicaciones (MINCOM) (3). Este centro como organización enfocada a contribuir al desarrollo de la industria, propone al ministerio normativas, técnicas, políticas, procedimientos y estándares para las entidades que desarrollan aplicaciones informáticas. Tiene como objeto

social, además, la responsabilidad de evaluar la conformidad con las normas y emitir certificación a productos informáticos nacionales o importados; brindar servicios de consultorías asociadas a la ingeniería y calidad de aplicaciones informáticas; controlar y auditar el uso de normativas técnicas, procedimientos, documentos estandarizados y buenas prácticas para el desarrollo de aplicaciones y sistemas informáticos en el país. También brinda servicios de formación en los temas de calidad e ingeniería de aplicaciones informáticas. CALISOFT en su estructura tiene el Departamento de Evaluación de Productos (DEPSW), cuyo objetivo fundamental es la realización de pruebas de liberación con la calidad requerida y la evaluación de la calidad de los productos de software solicitados por las empresas desarrolladoras como las empresas adquisidoras de los productos del país. El DEPSW evalúa la calidad del software a través de la NC ISO/IEC 9126-1 2005 y NC ISO/IEC 25000: 2011.

En la actualidad, este centro realiza el proceso de evaluación de la calidad de productos de software y el procesamiento de los datos se realiza de forma semiautomática, apoyado en hojas de cálculo en Excel. En estas hojas de cálculo, se introducen determinados elementos de medidas y las No Conformidades (NC) encontradas en las diferentes pruebas de software realizadas. A partir de ellos se obtienen el valor de las medidas, de ahí se calcula las subcaracterísticas, las características y el índice de calidad a partir del promedio. Una vez hecho este paso se procede a clasificar el software en aceptable, parcialmente aceptable o no aceptable según el valor del índice de calidad. Este proceso se realiza con cada uno de los productos de software. Se ordenan los índices de calidad de manera descendiente y el que mejor resultado posea (el primero de la lista ordenada) es el que será elegido.

Este proceso semiautomático implica atrasos en la información e inexactitud de los resultados. Además, esto provoca que solo se tenga en cuenta las medidas objetivas, despreciando las medidas subjetivas. También el promedio no es la medida óptima para obtener el índice de calidad, ya que no permite sopesar unas características más que otras según el objetivo de la evaluación, tampoco tiene en

cuenta las relaciones que existen entre características. Debido a estos problemas se dificulta el proceso de agregación de la información y la toma de decisión.

Existen múltiples métodos para agregar la información según representa la bibliografía consultada, capaces de solucionar problemas como el antes mencionado (4), (5), (6), (7). Los métodos que más se destacan son los métodos de decisión multicriterio (MCDM por sus siglas en inglés) ya sean manipulando datos crisp como números difusos. Se utiliza el AHP, ANP, PROMETHEE y TOPSIS, entre otros. También se proponen variaciones de los existentes como métodos híbridos donde combinan dos o más métodos aprovechando las ventajas y minimizando las desventajas.

Es una necesidad de CALISOFT realizar las evaluaciones usando diferentes técnicas de agregación de la información a través de los MCDM y comparar los diferentes rankings obtenidos por los métodos. Este proceso facilitaría la toma de decisión y se obtendrían índices de calidad con mayor precisión. Para ello es esencial el uso de un sistema automatizado que guie el proceso de agregación de la información por diferentes métodos y facilite la toma de decisión.

En el ámbito nacional no se cuenta con una aplicación informática que apoye este proceso según las fuentes consultadas. En el ámbito internacional los sistemas que se utilizan son sistemas propietarios de las empresas certificadoras de calidad de software. Estos software constituyen *know-how* de la empresa, que no es más que un conjunto de conocimientos producto de la información, la experiencia y el aprendizaje, que no pueden ser protegidos por una patente por no constituir una invención en el sentido estricto del término, pero que son determinantes del éxito comercial de una empresa, para su protección sólo cabe la fórmula del secreto (8), por lo que no se comparten, ni se tienen a disposición de otras empresas.

Hay un conjunto de herramientas informáticas que implementan los MCDM que se pueden utilizar para resolver problemas genéricos. CALISOFT para utilizar estas herramientas tiene que modelar su problemática y adaptar sus datos, lo que provocaría retrasos en la toma de decisión y la contratación o formación de especialistas en métodos multicriterio.

La revisión de la literatura científica acerca de la evaluación de la calidad de los productos de software, de conjunto con las entrevistas realizadas a directivos de CALISOFT ha permitido identificar los siguientes elementos que determinan la situación problemática:

- El proceso de evaluación de la calidad del producto de software resulta bastante costoso y complejo, lo que hace difícil implementarlo de forma manual.
- Se dificulta el proceso de agregación de la información a la hora de evaluar la calidad de productos de software y por ende la toma de decisión.
- La no existencia de plataformas web que integren varios métodos de decisión multicriterio para agregar la información resultante de las pruebas de software.
- No se comparan los resultados de la evaluación de la calidad del software, obtenidos por varios métodos de agregación, que facilitaría el proceso de toma de decisión.
- La poca flexibilidad de las plataformas existentes al adicionar nuevos métodos de agregación de la información.

Lo que conduce a enunciar el siguiente **problema de investigación**

> ¿Cómo contribuir a la toma de decisión en la evaluación de la calidad de productos de software con la ayuda de las Tecnologías de la Información y las Comunicaciones?

Teniendo en cuenta el problema de investigación se define como **objeto de estudio,** las herramientas de ayuda a la toma de decisión en el proceso de evaluación de la calidad de productos de software. Enmarcando el **campo de acción** en las aplicaciones web de apoyo a la toma de decisión en el proceso de evaluación de la calidad de productos de software en CALISOFT

A partir del problema de investigación se define como **objetivo general:**

> Desarrollar una aplicación web que ayude a la toma de decisión en la evaluación de la calidad de productos de software comparando diferentes métodos de agregación.

Para dar cumplimiento al objetivo general se definieron los siguientes **objetivos específicos:**

1. Caracterizar el proceso de evaluación de la calidad de productos de software como un proceso de toma de decisión, así como el análisis de las herramientas y los diferentes métodos de agregación de la información.
2. Implementar una aplicación web que ayude a la toma de decisión en la evaluación de la calidad de productos de software comparando diversos métodos de agregación de la información.
3. Validar el software haciendo uso de los métodos definidos en la investigación.

Para la resolución de la problemática anterior se utilizaron los **métodos de investigación científica**, tanto teóricos como empíricos.

Los métodos teóricos utilizados son:

- Analítico-Sintético: Su objetivo en la investigación es analizar las teorías, documentos, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

  Se analizaron diferentes documentaciones e informaciones expuestas en sitios oficiales, posibilitando la extracción de las características principales del proceso de evaluación de la calidad de productos de software que ayuda al proceso de toma de decisión, así como los métodos de decisión multicriterio que permitan agregar la información.
- Análisis Histórico-Lógico: Su objetivo en la investigación es constatar teóricamente cómo ha evolucionado un determinado fenómeno en un período de tiempo, en toda su trayectoria o en un fragmento temporal de la lógica de su desarrollo.

  Se constató la evolución de los sistemas de toma de decisión, así como sus métodos y características esenciales en los últimos cinco años.

Además, se empleó como método empírico la entrevista. Esta consiste en una conversación planificada para obtener información. Se realizaron entrevistas al jefe de departamento de pruebas de software de CALISOFT.

El documento se encuentra estructurado como sigue: introducción, tres capítulos que forman el cuerpo del documento, conclusiones, recomendaciones, referencias bibliográficas y los anexos.

El **primer capítulo** se titula “Breve análisis al problema de automatización de la evaluación de la calidad de software que ayude al proceso de toma de decisiones”, y tiene como objetivo principal analizar y presentar los elementos teóricos base de la investigación. Los conceptos esenciales relacionados con toma de decisión multicriterio. Además, posee una descripción de los métodos y metodologías que resuelven problemas de toma de decisión, así como una breve caracterización de sistemas existentes que dan solución al problema planteado. También destaca las herramientas, metodologías y lenguajes de programación que se van a utilizar en el desarrollo del sistema.

El **segundo capítulo** se titula “Análisis y diseño de la propuesta de solución”, su objetivo principal es presentar la propuesta web que da solución a problemas de toma de decisión multicriterio y sus fundamentos. Posee los elementos que caracterizan de la propuesta de solución, así como las características y funcionalidades del sistema a partir de los requisitos funcionales y no funcionales identificados. También posee los patrones de diseño, la arquitectura a utilizar en el sistema, en conjunto de otros elementos necesarios para desarrollar la solución propuesta.

El **tercer capítulo** se titula “Implementación y pruebas del sistema”, este consiste en presentar los resultados de la implementación y pruebas realizadas a la aplicación. Se describe el proceso de construcción de la aplicación web, así como las pruebas a realizar para validar su correcta elaboración y funcionamiento.

CAPÍTULO 1. Breve análisis al problema de automatización de la evaluación de la calidad de software que ayude al proceso de toma de decisiones

Este capítulo tiene como objetivo presentar, analizar y detallar los elementos teóricos que sustentan la presente investigación, para ello se realiza un estudio de los modelos de calidad de software, del proceso de evaluación de la calidad de productos de software existentes, así como los elementos a usar para agregar la información en el proceso de evaluación de la calidad, seleccionando los elementos que puedan ser empleados para dar solución al problema planteado. Se realiza un análisis de diferentes productos de software involucrados en este proceso.

1.1. Evaluación de la Calidad de productos de software

Las definiciones de los diferentes términos y aspectos que conforman la calidad, el proceso de evaluación de la calidad y los modelos se abordan en estándares. Un estándar es un documento que proporciona los requisitos, especificaciones, directrices o características que se pueden utilizar constantemente para asegurar que los materiales, productos, procesos y servicios son adecuados para su propósito (9). Un estándar sirve como modelo, patrón o referencia. Hay varias organizaciones que definen estándares, pero la más destacada es la Organización Internacional de Normalización (ISO de las siglas en inglés, *International Organization for Standardization*).

A partir del concepto de calidad dado por los autores principales de la Ingeniería de software y por los estándares mostrados en la introducción, se puede definir calidad de software como el grado en que el software sea capaz de cumplir los requisitos establecidos, los estándares especificados y de satisfacer las expectativas del cliente, cuando es usado bajo determinadas condiciones. Expresar la calidad en términos de cualidades y características es difícil, por esta razón es que surgen diversos modelos de calidad de software y estándares.

Un modelo de calidad de software es una representación de la calidad a partir de criterios o características. Esta descomposición permite satisfacer las necesidades

de los desarrolladores, las necesidades de los encargados de dar mantenimiento, de los clientes que compran el software y de los usuarios finales. Además, proporciona las bases para la especificación de los requisitos de calidad y la evaluación de la misma.

Es esencial valorar numéricamente la calidad para comparar, seleccionar y adquirir productos de software. Esto se logra en el proceso de evaluación de la calidad de productos de software. Este proceso se encuentra regido por las siguientes normas de estandarización:

- ISO 9126 y 14598.
- Familia ISO 25000.

1.1.1. Estándares y Modelos de Calidad

ISO 9126 y 14598

La Norma ISO 9126 que es un estándar internacional para la evaluación de la calidad del software está dividido en cuatro partes. Estas son: 1) modelo de calidad, 2) métricas externas, 3) métricas internas y 4) métricas de calidad de uso (Ver Figura 1.1).

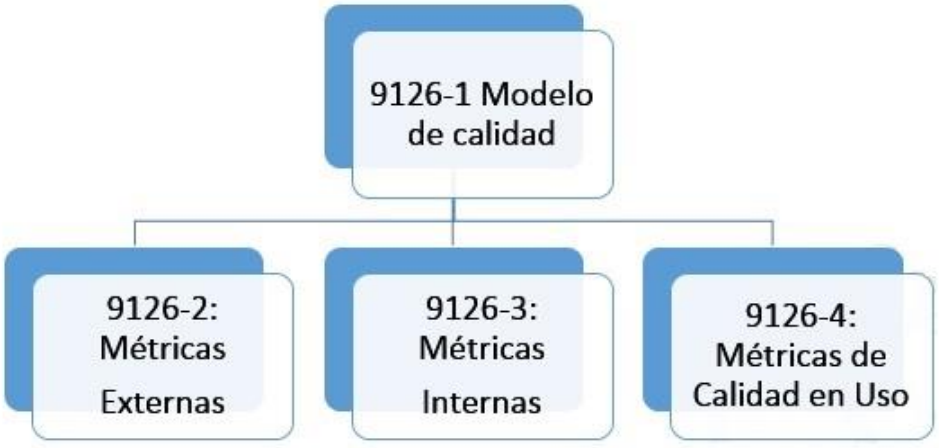


Figura 1.1: Norma ISO 9126.

El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas. (Ver Figura 1.2) (10)

Figura 1.2: Estructura de las características y subcaracterísticas de la Norma ISO 9126-1.

La ISO 14598 complementa la ISO 9126 ya que establece un marco de trabajo para evaluar la calidad de los productos de software proporcionando, además, métricas y requisitos para el proceso de evaluación de los mismos. En ella se definen y se describen las actividades necesarias para analizar los requisitos de evaluación, para especificar, diseñar y realizar acciones de evaluación y para concluir la evaluación de cualquier tipo de producto de software (11). En particular, es utilizada para aplicar los conceptos descritos en la norma ISO/ IEC 9126.

Mientras se utilizaban las dos series de normas ISO 9126 e ISO 14598, se identificaron problemas causados principalmente por los avances tecnológicos y otras cuestiones adicionales que podían mejorarlas (12). Además, la existencia de dos series de estándares complicaba la comprensión y la aplicación de estas normas. Por otra parte, las métricas estaban definidas en diferentes normas que había que buscar y relacionar. Por todo lo anterior, se decide unir todo en una nueva arquitectura, dando origen a una nueva familia de estándar la ISO 25000 (proyecto SQuaRE- *System and Software Quality Requirements and Evaluation*).

Serie ISO 25000

A partir de las normas anteriores surge la Norma ISO 25000 donde su objetivo general es organizar, enriquecer y unificar las series que cubren dos procesos principales: especificación de requisitos de calidad del software y evaluación de la calidad del software, soportada por el proceso de medición de calidad del software. Las características de calidad y sus mediciones asociadas pueden ser útiles no solamente para evaluar el producto software sino también para definir los requisitos de calidad.

Dentro de este último estándar se encuentran varias divisiones o normas, pero se hará referencia a la Norma ISO 25020 que recoge el modelo de calidad semejante a la Norma ISO 9126 que la sustituyó, y a la Norma ISO 25040 que recoge en si el procedimiento de evaluación de calidad de productos de software derivándose de la Norma ISO 14598 (Ver Figura 1.3).

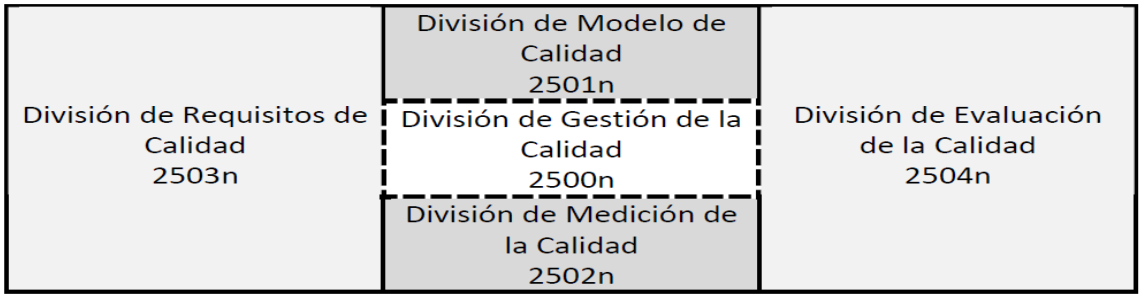


Figura 1.3: Organización de la Serie de Normas Internacionales SQuaRE. **(13)**

La Norma ISO 25020 proporciona una guía para la medición de las características de calidad. También contiene juegos de requisitos para la selección y construcción de medidas de calidad. Además, posee anexos informativos que abordan los siguientes temas: criterios para la selección de las medidas de calidad de software y elementos de medición de la calidad, lo que demuestra la validez predictiva y la evaluación de la seguridad de medición, y un formato de ejemplo para la documentación de las medidas de calidad de software. (Ver Figura 1.4) (14)

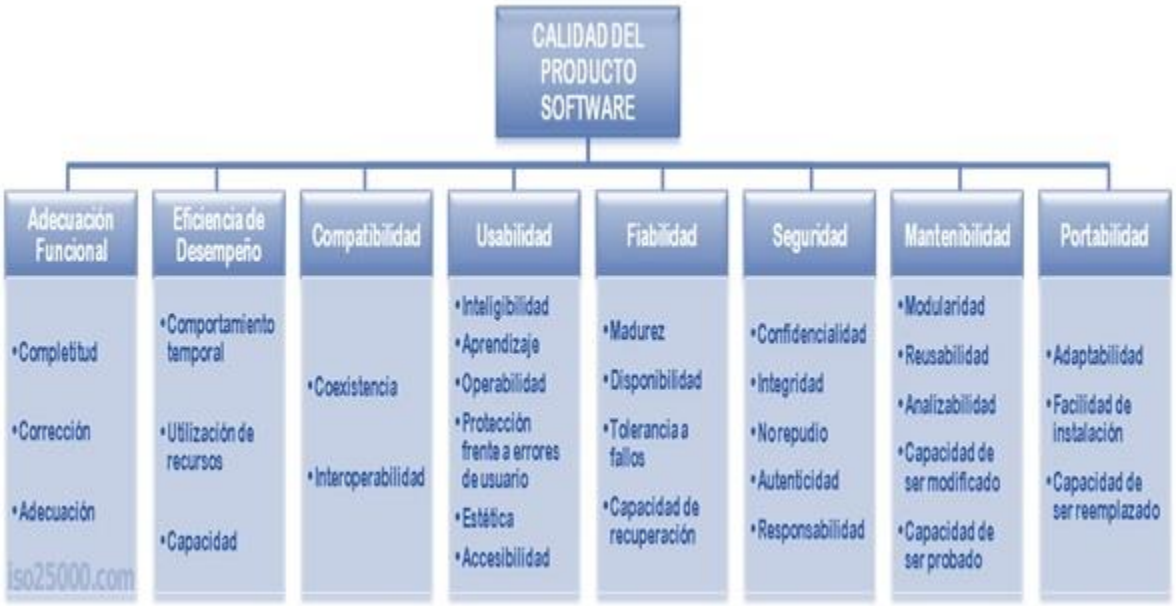


Figura 1.4: Medidas de calidad de software y elementos de medición de la calidad.

La Norma ISO 25040 define el proceso para llevar a cabo la evaluación del producto de software. Dicho proceso de evaluación consta de un total de cinco actividades (Ver Figura 1.5).

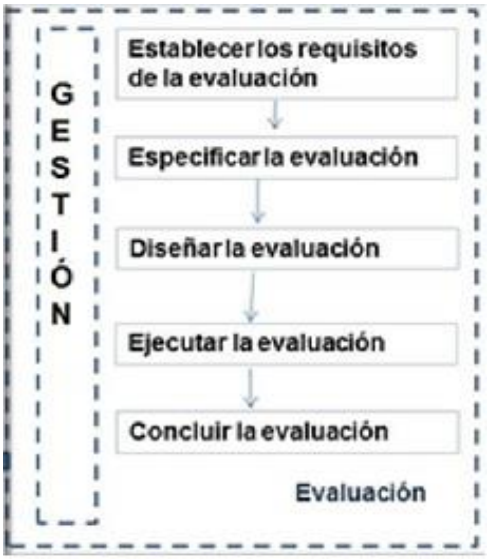


Figura 1.5: Proceso de evaluación según ISO 25040 **(15)**.

Todas las normas mencionadas anteriormente fueron creadas con el fin de resolver un problema: “la evaluación de la calidad de productos de software”.

El modelo de calidad que usa actualmente CALISOFT tiene una estructura jerárquica integral, como se aprecia en la Figura 1.6. Las características poseen a

su vez subcaracterísticas y estas se miden a través de atributos de calidad, que se obtienen a través de medidas de calidad. Para obtener un índice de calidad de los productos, basta con obtener el valor de las subcaracterísticas a partir de las medidas, obtener el valor de las características a partir de las subcaracterísticas o las medidas y obtener el índice de calidad a partir del valor de las características.

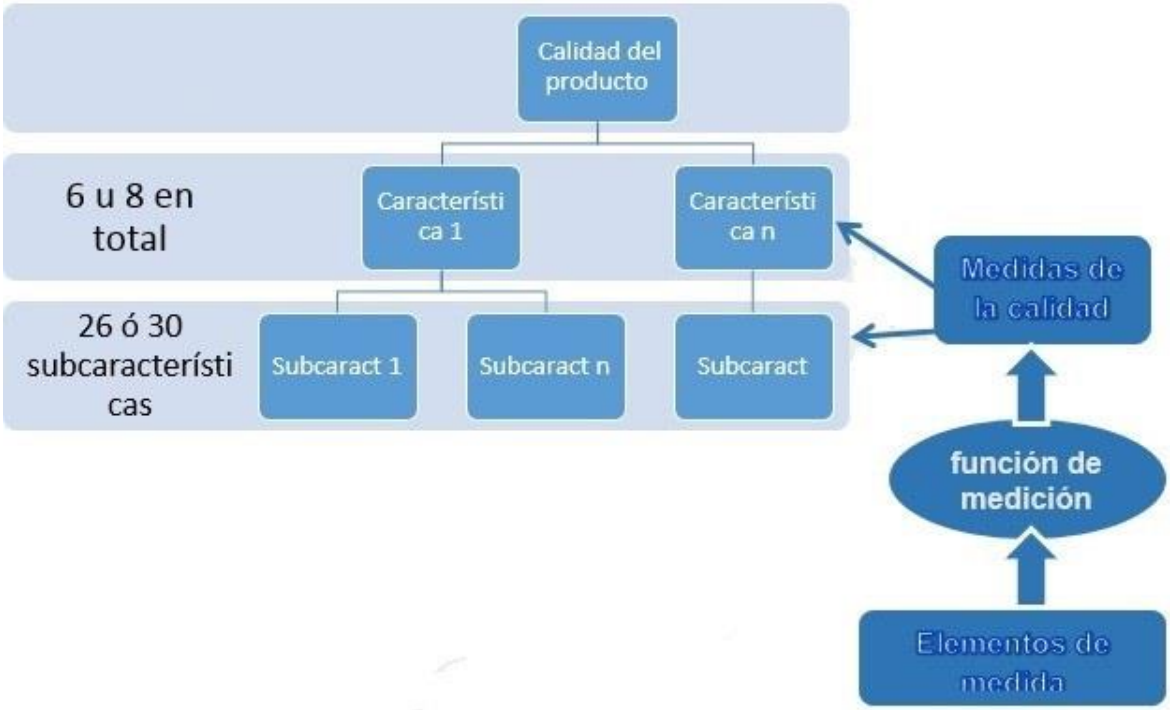


Figura 1.6: Estructura de los modelos de calidad (ISO 25000).

CALISOFT usa como modelo de calidad un intermedio entre la norma ISO 9126 y la ISO 25010, que contiene las siguientes características:

- Funcionalidad
- Seguridad
- Portabilidad
- Usabilidad
- Eficiencia
- Confiabilidad

Estas características poseen a su vez subcaracterísticas, y estas asociadas a medidas de calidad, tributan a las características globales.

A partir del análisis de las normas, de los conceptos generales y de los modelos de calidad se logra la caracterización de este proceso. Si se compara el proceso de evaluación de la calidad de los productos de software con el proceso de toma de decisión (TD). (Ver figura 1.7). Se puede observar que ambos procesos tienen actividades en común. La primera parte de estructuración del problema en el proceso de TD se determinan los criterios y las alternativas, esto es semejante a lo que se realiza en las dos primeras actividades del proceso de evaluación donde se determina el modelo de calidad (múltiples criterios organizados jerárquicamente) y los productos a evaluar. En los dos procesos se evalúan las alternativas y concluyen eligiendo una opción o representando un ranking. Por lo anterior, se puede concluir que la evaluación de la calidad de productos de software se puede modelar como un problema de toma de decisión.

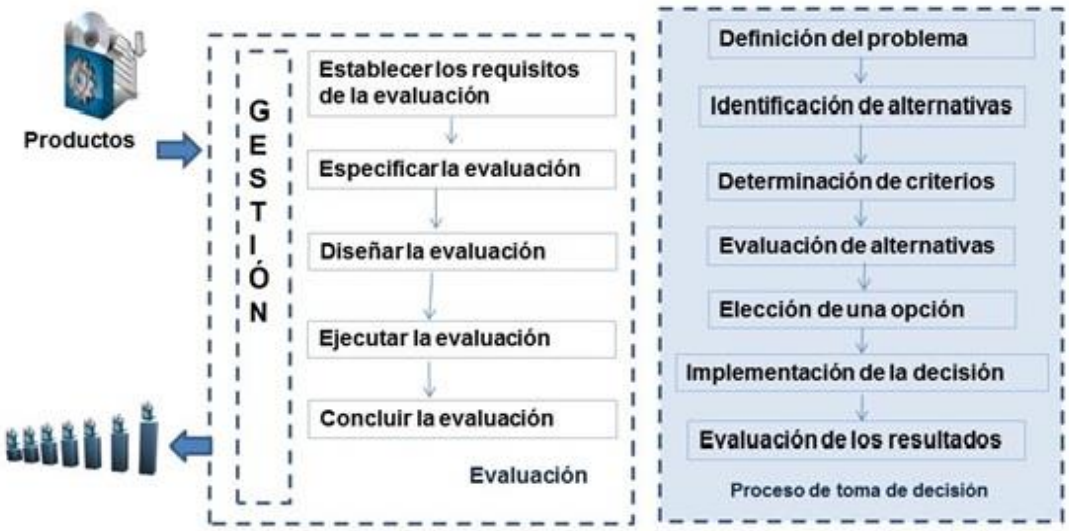


Figura 1.7: Proceso de evaluación de la calidad de productos de software.

La toma de decisiones (TD) es una actividad cotidiana de la vida humana. A diario las personas se encuentran en situaciones en las que deben decidir entre alternativas u opciones y seleccionar las que cumplan con el mayor número de determinadas condiciones. La toma de decisión es el proceso mediante el cual se toma una elección entre varias opciones con el fin de resolver diferentes situaciones de la vida en determinados contextos: laboral, familiar, personal, sentimental o empresarial (utilizando metodologías cuantitativas que brinda la

administración). La misma consiste, básicamente, en elegir una opción entre las disponibles, a los efectos de resolver un problema actual o potencial (aun cuando no se evidencie un conflicto latente). Esta actividad humana es primordial en la actividad industrial. La industria del software no es una excepción. Como se observa en la figura 1.7, en el caso de la evaluación de la calidad de software, la Norma ISO 25040 define un conjunto de actividades para el desarrollo del proceso de evaluación de productos de software; estas actividades son muy semejantes a los pasos definidos por el proceso de toma de decisión multicriterio. Esta semejanza permite representar el proceso de evaluación de la calidad como un problema de toma de decisión multicriterio.

1.2. Herramientas operativas para resolver problemas de toma de decisión

1.2.1. Toma de decisión multicriterio

Un individuo o un grupo de individuos para tomar decisiones frente a un conflicto o problema se plantean un conjunto bien definido de alternativas. Además, establecen un conjunto de criterios para valorar las alternativas, establecen también un conjunto de preferencias en cuanto a los criterios y selecciona una alternativa o establece un orden entre las alternativas. Todo proceso de decisión transcurre en un contexto que se denomina ambiente o entorno. Como se puede observar en este proceso participan varios elementos: decisores, alternativas, criterios de decisiones y pesos asociados a dichos criterios. Los decisores son el conjunto de individuos que tienen la responsabilidad de tomar la decisión. Por su parte los criterios de decisión son los parámetros que se utilizan para valorar las preferencias del decisor en cuanto a las alternativas, son elementos de referencia en base a los cuales se realiza la decisión. Los pesos o ponderaciones son las medidas de importancia relativa que los criterios tienen para el decisor. Se definen alternativas como el conjunto finito de soluciones, acciones, opciones posibles que hay que analizar durante el proceso de toma de decisión que se considere.

Un problema de decisión de forma general parte de un conjunto de **alternativas** se designa por $A = \{A_1, A_2..., A_m\}$ donde A_i (i=1..., m), que son evaluadas para un conjunto de criterios que son los **criterios de decisión** $C = \{C_1, C_2..., C_n\}$ donde Cj

(j= 1..., n) n>=2. Asociado a los criterios, se asigna un **vector de pesos** W= {w_1, w_2..., w_n}. Este valor se asigna por los métodos: método de asignación directa y método del autovector (*eigenvector*). Estos elementos conforman una matriz, denominada matriz de decisión (Ver figura 1.8).

	w_1	w_2		w_n
	C_1	C_2	...	C_n
A_1	x_{11}	x_{12}	...	x_{1n}
A_2	x_{21}	x_{22}	...	x_{2n}
A_m	x_{m1}	x_{m2}	...	x_{mn}

Figura 1.8: Matriz de decisión.

Los diferentes métodos de solución actúan sobre la matriz para valorar las alternativas, a través de una operatoria dada y orientan al decisor sobre cual o cuales son las mejores alternativas. De esta valoración se devuelve un valor asociado a cada alternativa, por el cual son ordenadas finalmente. $\{\boldsymbol{A_i}\}$ **donde** $\mathbf{1 \leq i \leq m}$, $\mathbf{A_i^1 \succcurlyeq A_i^2 \succcurlyeq \cdots \succcurlyeq A_i^m}$

En el proceso de evaluación de la calidad de software, al tratarlo como un proceso de toma de decisión multicriterio se aplican los métodos más utilizados para solucionar este tipo de problema.

Para una mayor comprensión y un mejor trabajo con los problemas multicriterio, estos métodos se clasifican en dos categorías: los de programación múltiple objetiva y los de evaluación multicriterio. En el presente trabajo se hace referencia a la segunda categoría antes mencionada. Muchos de los métodos empleados en la actualidad para solucionar problemas de toma de decisión se agrupan en dos escuelas: la escuela europea y la escuela americana; a la primera pertenecen algunos métodos como: ELECTREE, PROMETHEE y MACEBTH, luego se hará referencia a algunas aplicaciones que utilizan estos métodos. También agregar que a la segunda escuela pertenecen métodos tales como: AHP y ANP. No obstante, existen también otros métodos que no se encuentran dentro de estas

escuelas, pero se pueden agrupar como otros métodos de decisión multicriterio, ese es el caso del método PAPRIKA, TOPSIS.

A continuación, se explican los métodos más utilizados.

1.2.2. Métodos de decisión multicriterio

Método AHP

El Proceso de Análisis Jerárquico (AHP[1]) es un método creado por el Dr. Thomas Saaty (16) que se basa en descomponer jerárquicamente el problema a tratar en sus componentes. En el más alto nivel de la jerarquía se ubica el objetivo a alcanzar. El segundo nivel está constituido por los sub-objetivos y criterios que permitirán evaluar el grado de logro de los mismos por las diferentes alternativas que están en el tercer nivel. De esta manera las alternativas serán ordenadas jerárquicamente a través de los pesos globales calculados mediante el método (17).

Una aplicación útil del método es que posibilita determinar los coeficientes de la matriz de decisión cuando no se conocen sus valores, ya sea por falta de datos o por incertidumbre, dado que permite determinarlos sobre la base de las preferencias del centro decisor y en función de los criterios (18).

AHP consta del siguiente algoritmo, en el cual se repite para cada criterio los pasos del 1 al 4:

Paso 1: Se debe crear la Matriz de Comparación por Pares de alternativas para cada uno de los criterios y establecer el peso de importancia relativa entre cada par de alternativas consideradas. El peso de importancia se establece a partir de una escala del 1 al 9, donde 1 es muy poco importante y 9 es extremadamente importante. Se aplica un peso recíproco cuando la segunda alternativa es más importante que la primera (ejemplo: $1/9$, $1/3$). El valor 1 siempre es asignado a la comparación de una alternativa con sí misma.

Paso 2: Se crea la Matriz Normalizada, donde se divide cada número de la Matriz de Comparación por Pares por la suma total de su columna.

[1] Del inglés *Analytic Hierarchy Process*

Paso 3: Se determina el Vector de Prioridad para el criterio calculando el promedio de cada fila de la Matriz Normalizada. Este promedio por fila representa el Valor de Prioridad de las alternativas con respecto al criterio seleccionado.

Paso 4: Se deben reconsiderar las opiniones y juicios expresados por los expertos. Para esto es necesario determinar la Consistencia de las opiniones utilizadas en la Matriz de Comparación por Pares utilizando la Razón de Consistencia RC. Una $RC \leq 0.10$ es considerada aceptable, en caso de que $RC > 0.10$ se deben reconsiderar los juicios y opiniones.

Para determinar el valor de RC se debe realizar para cada fila de la Matriz de Comparación por Pares de alternativas, una suma ponderada en base a la suma del producto de cada celda por la prioridad de cada alternativa correspondiente. Para cada fila de la matriz se divide la suma ponderada por la prioridad de su alternativa correspondiente. Luego se determina la media del cálculo realizado anteriormente n_{max} y se calcula el Índice de Consistencia para cada alternativa $IC = \frac{n_{max}-n}{n-1}$. Seguido de esto se calcula el Índice de Aleatoriedad $IA = \frac{1.98(n-2)}{n}$ y finalmente se obtiene $RC = \frac{IC}{IA}$.

Paso 5: Luego de haber realizado los pasos anteriores para cada criterio, los resultados obtenidos en el paso 3 son resumidos en una Matriz de Prioridad, listando las alternativas por fila y los criterios por columna.

Paso 6: Se confecciona una Matriz de Comparación por Pares de criterios de manera similar a como se hizo para las alternativas en los pasos del 1 al 3.

Paso 7: Se desarrolla un Vector de Prioridad Global multiplicando el Vector de Prioridad de los criterios obtenido en el paso 6 por la Matriz de Prioridad de las alternativas obtenido en el paso 5.

Paso 8: Finalmente se selecciona la alternativa que tenga un mayor valor en el Vector de Prioridad Global.

Método ANP

A diferencia del AHP, que resulta más práctico en problemas de toma de decisión donde los niveles jerárquicos son independientes unos de otros, el Proceso Analítico en Redes (ANP[2]) es usado en problemas de toma de decisión donde existen factores que dependen de otros factores en el mismo nivel jerárquico u otros niveles. Este método, inventado por el Dr. Thomas Saaty, permite el uso de muchos factores cualitativos, y provee además el nivel de importancia de cada criterio o sub-criterio, de manera que cada decisor o experto pueda determinar qué criterio es más relevante (19).

ANP es una derivación de AHP y consta de los siguientes pasos:

Paso 1: Modelar el problema de decisión como una red identificando los criterios y alternativas, agrupándolos en componentes y determinando las relaciones de interdependencia entre ellos.

Paso 2: Realizar comparaciones por pares entre cada elemento.

Paso 3: Construir una supermatriz no ponderada SNP con los vectores de pesos de importancia relativa de los elementos.

Paso 4: Realizar comparaciones por pares entre componentes.

Paso 5: Ponderar los bloques de la supermatriz no ponderada SNP, mediante los pesos correspondientes de los componentes para transformarla en la supermatriz ponderada SP.

Paso 6: Si es necesario, normalizar la supermatriz ponderada SP, dividiendo cada valor por la suma de las columnas. De esta forma se obtiene una supermatriz ponderada estocástica SPE por columnas, es decir, cuyas columnas sumen la unidad.

Paso 7: Elevar la supermatriz ponderada estocástica SPE a potencias sucesivas hasta que sus entradas converjan y permanezcan estables obteniendo la supermatriz límite SL.

Paso 8: Finalmente se selecciona la alternativa que tenga mayor valor en la supermatriz límite SL.

[2] Del inglés *Analytic Network Process*

La selección de personal es un proceso de la Gestión de Recursos Humanos en la que intervienen múltiples factores y criterios que están relacionados unos con otros, el método ANP es muy usado en este campo debido a que proporciona facilidades en este tipo de problemas (19).

ANP es uno de los métodos que más se destaca, ya que tiene en cuenta la interdependencia entre criterios, aunque es difícil de aplicar porque sus resultados dependen de la estructura de grafo que se define, además por la cantidad de preguntas que se generan por la comparación a pares y algunas de ellas no son intuitivas para los expertos.

Método TOPSIS

El método TOPSIS[3] tiene como objetivo identificar las soluciones dentro de un conjunto finito de alternativas y buscar la solución a partir de la definición de una solución ideal positiva y una solución ideal negativa. El principio básico del método es encontrar la alternativa que tenga la menor distancia a la solución ideal positiva y la mayor distancia de la solución ideal negativa, para lo cual se define un índice de similitud que se construye combinando la proximidad y la lejanía a los ideales planteados (20).

Este método no fue creado bajo el paradigma de la lógica difusa pero se han definido extensiones (TOPSIS Difuso) que permiten el tratamiento de la incertidumbre (21).

TOPSIS parte de la consolidación de la información proporcionada por los expertos E, a cada uno de ellos se le solicita una matriz de decisión $\widehat{MD}$. También se le pide que ofrezca su valoración de la importancia de cada criterio W (22).

El algoritmo del método es el siguiente:

Paso 1: Se normaliza la matriz de decisión $\widehat{MD}$, obteniéndose la matriz normalizada, conocida como $\hat{R} = [\widehat{r_{i,j}}]$, donde $\widehat{r_{i,j}} = \frac{1}{cmax(\hat{X})} \odot \widehat{x_{i,j}}$.

Paso 2: Se multiplica $\hat{R}$ por el vector de pesos W para obtener la matriz de decisión normalizada y ponderada $\hat{P} = [\widehat{p_{i,j}}]$, donde $\widehat{p_{i,j}} = \widehat{r_{i,j}} \otimes w_i$.

[3] Del inglés *Technique for Order Preference by Similarity to Ideal Solution*

Paso 3: A partir de la matriz $\widehat{\mathrm{P}}$ se calcula la solución ideal positiva $A^+ = (p_1{}^+, p_2{}^+, \dots, p_n{}^+)$ y la solución ideal negativa $A^- = (p_1{}^-, p_2{}^-, \dots, p_n{}^-)$. El cálculo de $p_j{}^+$ y de $p_j{}^-$ se realiza según sea un costo o un beneficio. Para $p_j{}^+ = \left({}_i^{max} P_{i,j}, j \in J_1;\ {}_i^{min} P_{i,j}, j \in J_2; \right)$ y para $p_j{}^- = \left({}_i^{min} P_{i,j}, j \in J_1;\ {}_i^{max} P_{i,j}, j \in J_2; \right)$.

Paso 4: Se calcula la distancia de cada alternativa a los ideales A^+ y A^-, para ello se pueden utilizar la distancia de Euclides, de Minkowsky, de Haming, de Tchebyschev, el método del vértice, la distancia Camberra, y Mahalanobis, entre otras muchas de las que han sido definidas.

Paso 5: Finalmente se escoge la alternativa para la cual la formula $\varepsilon = \frac{d_i{}^-}{d_i{}^+ + d_i{}^-}$ devuelve el valor mayor, $d_i{}^+$ y $d_i{}^-$ son las distancias del candidato respecto a los ideales A^+ y A^-. Esta fórmula representa el concepto central de TOPSIS donde la alternativa preferida es la que se encuentra a la menor distancia de A^+ y a la mayor distancia de A^-.

Para definir la solución ideal positiva se deben establecer los puntajes ideales de cada uno de los atributos o criterios, por lo general inalcanzables para cualquier alternativa por lo que este método trata de encontrar siempre la alternativa que más se le acerca a este ideal positivo. De manera opuesta, para definir la solución ideal negativa se establecen los peores puntajes de cada uno de los criterios, tratando siempre de encontrar una alternativa que se aleje lo más posible de este ideal negativo.

El método TOPSIS se ha utilizado en diferentes áreas y su uso en la evaluación de la calidad de software es idóneo si se tiene conocimiento de las características que necesita un software (ideal positivo) o de las características que no puede tener el mismo (ideal negativo) para cumplir con las exigencias del cliente (23).

Hay que señalar de estos métodos que ninguno tiene en cuenta la interdependencia entre criterios, excepto el ANP. Pero el método ANP es difícil de aplicar debido a la cantidad de preguntas que se realizan durante las comparaciones a pares.

El uso de los mapas cognitivos difusos puede ser una alternativa para el tratamiento de la interdependencia, es una vía alternativa a la aplicación de ANP.

Mapas cognitivos difusos

En la Inteligencia Artificial los mapas cognitivos son redes capaces de aprender, codificar y descodificar conocimientos con respecto a eventos casuales y la forma en la que es activada. Modelar mapas cognoscitivos utilizando lógica difusa (MCD) parece natural, debido a la inherente incertidumbre que se encuentra en las bases de datos y conocimiento del mundo real. [6]

Los mapas cognitivos difusos fueron ideados por Kosko (1986) como una extensión de los mapas cognitivos de Axelrod (1976), constituyen una estructura de grafo difuso utilizado para representar razonamiento causal; su aplicación resulta recomendable para los dominios donde los conceptos y las relaciones son fundamentalmente difusos como la política, la historia, la planificación estratégica, etcétera. (24)

Los mapas cognitivos difusos (MCD), son grafos difusos dirigidos con retroalimentación para representar casualidad, combinan herramientas teóricas de los mapas cognitivos, la lógica difusa, las redes neuronales, las redes semánticas, los sistemas expertos y los sistemas dinámicos no lineales. Son una herramienta de modelado cualitativa, que se basa en los conocimientos y la experiencia de los expertos Los nodos representan los conceptos y los arcos pesados representan relaciones causales entre los conceptos. En los MCD existen tres posibles tipos de relaciones entre conceptos: relación positiva, relación negativa o la no existencia de relaciones. Un MCD formado por *n* conceptos, se representa en una matriz de *NxN* conocida como matriz de adyacencia. Esta matriz se obtiene a partir de los valores asignados a los arcos.

Estos mapas se pueden usar para representar la interdependencia que existen entre los diferentes criterios, que puede ser directa, inversa o no tener relación. La relación directa se presenta cuando al aumentar un criterio, aumenta también el criterio sobre el cual él influye. Por ejemplo: al aumentar la confiabilidad, aumenta la funcionalidad. La relación inversa es cuando al aumentar un criterio, el criterio dependiente disminuye. Por ejemplo, al aumentar la eficiencia, la usabilidad disminuye.

1.2.3. Herramientas de software que implementan los MCDM

Se analizaron sistemas que se usan en el proceso de agregación de la información en la evaluación de productos de software y sistema general para cualquier problema de toma de decisiones multicriterio.

A partir de las investigaciones realizadas a través de los documentos y los sitios oficiales existentes de los productos que se mencionan a continuación se pudo analizar y resumir brevemente las características de los mismos, atendiendo a los MCDM que implementan, el tipo de licencia y el dominio en el que se desenvuelven estos software.

Software que implementan MCDM

Software MPC 2.0: Es un software diseñado para facilitar la aplicación de la metodología de toma de decisiones AHP (*Analytic Hierarchy Process*), basada en la comparación por pares. El programa resulta especialmente útil en las decisiones en las que sea necesario considerar numerosos y diferentes tipos de criterios (cuantificables o no) y/o muchas posibles alternativas, así como también cuando sea preciso tener en cuenta diferentes repeticiones de la decisión de uno o diferentes usuarios. (25)

1000Minds: Se basa en un método de toma de decisiones denominado 'PAPRIKA'. PAPRIKA es un acrónimo de ‘’***P****otentially* ***A****ll* ***P****airwise* ***R****an****K****ings of all possible* ***A****lternatives*’'(Potencialmente todas las clasificaciones por parejas de todas las alternativas posibles). El método PAPRIKA está patentado en tres países: Estados Unidos, Nueva Zelanda, Australia. Este software pertenece a la línea de apoyo a la toma de decisión desarrollado por *100Minds Ltd.* en el año 2002. El sistema está disponible en el idioma inglés solamente y es una herramienta con licencia privativa que implementa el método PAPRIKA[4]. Este DSS es utilizado mayormente para priorizar o elegir entre alternativas en situaciones donde se necesita considerar varios objetivos o criterios simultáneamente (26)

[4] Del inglés *Potentially all pairwise rankings of all possible alternatives*

Super Decisions: El software Super Decisions es un programa comercial que se utiliza para resolver problemas de Decisión de Multicriterio. Incluye la solución de problemas de Procesos de Análisis Jerárquico (AHP) y los del tipo Proceso Analítico de Red (ANP). (27)

Expert Choice: Es un software para la toma de decisiones, implementa el Proceso Jerárquico Analítico (AHP, *Analytic Hierarchy Process*). Este sistema de apoyo a la toma de decisión fue desarrollado en 1983 por Thomas Saaty y Ernest Forman. Su licencia es privativa (26).

MultiDecision PAAT: Este sistema, desarrollado en el año 2013, es una herramienta web que implementa los métodos PROMETHEE, AHP, ANP y TOPSIS. No está diseñado para problemas que presenten incertidumbre, y no permite la incorporación de nuevos métodos de solución de problemas de toma de decisión (28), (29), (30).

Decision Making Helper: Es una herramienta orientada a ayudar a sus usuarios a tomar las decisiones más racionales y coherentes posibles de una lista de posibilidades que tendremos que facilitar previamente. El programa corre sobre todas las versiones de Windows y está disponible en los lenguajes inglés, alemán y francés. Los costos para una licencia de usuario son USD 25 o Euro 20. (31)

WinQSB: Es una herramienta cuyo objetivo es facilitar la labor en la toma de decisiones empresariales. El programa está dividido en una serie de módulos que ayudarán en los distintos tipos de tomas de decisiones. Así, tiene uno dedicado a la para resolver problemas lineales de objetivos definidos, otro para la decisión de análisis, e incluso uno de ellos especialmente dedicado al modelo oculto de Markov. (32)

Tabla 1.Sistemas para la solución a problemas de toma de decisión

Software o Aplicación	Métodos	Privativo o NO	Objetivos

Software MPC 2.0	AHP	SI	General
1000Minds	PAPRIKA	SI	General
Super Decisions	AHP,ANP	SI	General
Expert Choice	AHP	SI	General
MultiDecision PAAT	AHP, ANP, TOPSIS; PROMETHEE	SI	Especifico
Decision Making Helper	AHP.ANP, ELECTRE, PAPRIKA, PROMETHEE, AIRM,MAGIQ, MACBETH	SI	General
WinQSB		SI	General

Los sistemas informáticos mencionados anteriormente además de ser privativos se utilizan en dominios generales o específicos como la medicina, la agricultura, la política y el sector empresarial. Para usarlos en la evaluación de la calidad de productos de software, se hace necesaria la modelación del problema por especialistas y adaptar los datos de las medidas de la calidad. Por estas razones, se ve la necesidad de crear una herramienta que se adapte al proceso de evaluación de calidad de productos de software y que sea capaz de manejar los datos generados en el proceso.

1.2.4. Herramientas, Metodologías y Lenguajes de Desarrollo

Para el desarrollo de la solución informática, el cliente requería de una aplicación web sobre la plataforma de java. A partir de estos requerimientos se seleccionaron las siguientes herramientas de desarrollo:

Netbeans v8.0.1: Para el desarrollo del sistema se seleccionó como IDE Netbeans en su versión 8.0.1 debido a que es un entorno de desarrollo gratuito y de código abierto. Además, permite el uso de varias tecnologías de desarrollo tanto para escritorio, como aplicaciones web, o para dispositivos móviles. También brinda soporte a las siguientes tecnologías, entre otras: Java**,** PHP**,** Groovy**,** C/C++**,** HTML5. Además, puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS. Posee asistentes para la creación y configuración de un proyecto, tiene un buen editor de código, multilenguaje. También simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario. Además, permite el acceso a base de datos como Oracle y MySql permitiendo ver las tablas, realizar consultas y modificar la base de datos. (33)

Visual Paradigm: Es una herramienta CASE: Ingeniería de Software Asistida por Computación. Esta herramienta brinda un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Ha sido creada para soportar el ciclo de vida completo del proceso de desarrollo del software haciendo uso de los diferentes tipos de diagramas que permite realizar. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (Community Edition, ya que existe la Enterprise, Professional, etc). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. (34)

PgAdmin III: Es una aplicación de diseño y manejo de bases de datos para su uso con PostgreSQL. La aplicación se puede utilizar para manejar PostgreSQL 7.3

y versiones superiores, además funciona sobre casi todas las plataformas. Este software fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de la sintaxis SQL, un editor de código del lado del servidor, un agente para la programación de tareas «SQL/batch/shell», soporte para el motor de replicación Slony-I y mucho más. La conexión del servidor se puede realizar mediante TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede ser cifrado mediante SSL por seguridad. No se requieren controladores adicionales para comunicarse con la base de datos del servidor. (35)

Teniendo en cuenta las herramientas seleccionadas para el desarrollo del sistema, se seleccionaron los siguientes lenguajes de desarrollo:

Lenguaje UML: UML son las siglas de *"Unified Modeling Language"* o "Lenguaje Unificado de Modelado". Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos). (36)

Lenguaje JavaScript: Es un lenguaje de programación interpretado, lo que significa que no necesita ser compilado. Proviene del Java y se utiliza principalmente para la creación de páginas web. El JavaScript es una mezcla entre el Java y el HTML. Su creador fue Brendan Eich. Al principio se le llamo Mocha, más tarde LiveScript, hasta que en el año 1995 se le llamo JavaScript. (37)

Lenguaje Java: Es un lenguaje orientado a objeto, de plataforma independiente. Este fue creado por la compañía Sun Microsystems para el desarrollo de páginas web como idea originaria. La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como de aplicaciones

distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar. (38)

Lenguaje HTML5: Es un **lenguaje markup** (de hecho, las siglas de HTML significan *Hyper Text Markup Language*) usado para estructurar y presentar el contenido para la web. Es uno de los elementos esenciales para el funcionamiento de los sitios. Esta quinta revisión del estándar que fue creada en 1990. A finales del año pasado, la W3C la recomendó para transformarse en el estándar a ser usado en el desarrollo de proyectos venideros. (39)

Lenguaje SQL: SQL (*Structured Query Language*) es un lenguaje de programación estándar e interactivo para la obtención de información desde una base de datos y para actualizarla. Aunque SQL es a la vez un ANSI y una norma ISO, muchos productos de bases de datos soportan SQL con extensiones propietarias al lenguaje estándar. Las consultas toman la forma de un lenguaje de comandos que permite seleccionar, insertar, actualizar, averiguar la ubicación de los datos, y más. (40)

Por otra parte, el equipo de desarrollo encargado de implementar la aplicación es un equipo pequeño que tiene muy buena comunicación con el cliente, por lo que el proceso de desarrollo será iterativo e incremental. Teniendo en cuenta estas particularidades se seleccionó como metodología de desarrollo la Programación Extrema por sus siglas en inglés XP.

Metodología XP

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. En la misma se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además, detallan

la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (41)

Existen numerosas metodologías que con el pasar del tiempo se han ido adaptando a las características del producto informático a desarrollar. Un conjunto de estas metodologías son llamadas metodologías tradicionales debido a que priorizan el control del proceso, establecen las actividades involucradas, los diferentes artefactos que se deben producir y las herramientas a utilizar para producir los mismos. A parte de este grupo de metodologías también se pueden encontrar las llamadas metodologías ágiles, debido a que se enfoca más en la colaboración con el cliente y en el proceso de desarrollo incremental del producto basado en iteraciones de corto plazo. Las metodologías ágiles han mostrado una alta efectividad en proyectos con requerimientos muy cambiantes y en aquellos casos en los que se exige reducir notablemente los tiempos de desarrollo, pero sin descuidar la alta calidad del producto (42).

Características de la Metodología XP

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. XP es una metodología ágil para pequeños y medianos equipos, desarrollando software cuando los requerimientos son ambiguos o rápidamente cambiantes. Su ciclo de vida se basa en seis fases: Exploración, Planificación, Iteración, Producción, Mantenimiento y Muerte. Además, consta de cuatro fases fundamentales: Planeación, Diseño, Desarrollo y Pruebas. (43)

Planeación: La planeación es la etapa inicial de todo proyecto en XP. En este punto se comienza a interactuar con el cliente y el resto del grupo de desarrollo

para descubrir los requerimientos del sistema. En este punto se identifican el número y tamaño de las iteraciones al igual que se plantean ajustes necesarios a la metodología según las características del proyecto.

Diseño: En XP solo se diseñan aquellas historias de usuario que el cliente ha seleccionado para la iteración actual por dos motivos: por un lado, se considera que no es posible tener un diseño completo del sistema y sin errores desde el principio. El segundo motivo es que, dada la naturaleza cambiante del proyecto, el hacer un diseño muy extenso en las fases iniciales el proyecto para luego modificarlo, se considera un desperdicio de tiempo.

Desarrollo: El desarrollo es un proceso que se realiza en forma paralela con el diseño y la cual está sujeta a varias observaciones por parte de XP consideradas controversiales por algunos expertos tales como la rotación de los programadores o la programación en parejas.

Pruebas: Cuando se tienen bien implementadas las pruebas no habrá temor de modificar el código del otro programador en el sentido que, si se daña alguna sección, las pruebas mostrarán el error y permitirán encontrarlo. Uno de los elementos que podría obstaculizar que un programador cambie una sección de código funcional es precisamente hacer que este deje de funcionar. Si se tiene un grupo de pruebas que garantice su buen funcionamiento, este temor se mitiga en gran medida.

Se decidió hacer uso de esta metodología para el desarrollo de la aplicación debido a las ventajas que brindan sus características. Alguna de estas es la disminución del tiempo de desarrollo sin que se afecte la calidad del producto. Además, genera poca cantidad de artefactos y se encuentra enfocada principalmente al código.

Consideraciones parciales

A partir del análisis y el estudio realizado a los diferentes elementos necesarios para modelar el problema, se llegó a las consideraciones siguientes:

- La Norma de calidad 9126 que recoge el proceso de especificación de los requisitos de calidad y la Norma 14598 que encierra el proceso de evaluación de la calidad de software se encuentran unificadas y enriquecidas en la Norma ISO 2500 surgiendo la misma a partir de las anteriores, posibilitando la creación de un marco de trabajo integrado para la nueva solución.
- A partir de las diferentes literaturas consultadas, el estudio del proceso de evaluación de la calidad de software en CALISOFT y el estudio de las normas de calidad empleadas en dicho centro se coincide con otros autores de que el proceso de evaluación de la calidad de software se puede representar como un problema de toma de decisiones multicriterio.
- Los métodos de toma de decisiones multicriterio como AHP, ANP, TOPSIS y el método de agregación Suma Ponderada permiten tomar elecciones entre diferentes opciones o situaciones de la vida utilizando metodologías cuantitativas.
- Los sistemas informáticos que permiten dar solución a problemas de toma de decisión multicriterio en su mayoría son sistemas privativos que se enmarcan en áreas específicas como la administración del medio ambiente y la agricultura, la medicina, la política y el sector empresarial; sin embargo, ninguno integra los principios y características necesarios para llevar a cabo las actividades relacionadas al proceso de evaluación de la calidad de software.
- Teniendo en cuenta los requerimientos del cliente, las particularidades del equipo de desarrollo, el análisis de las diferentes herramientas, tecnologías y lenguajes de programación; el desarrollo de la propuesta de solución requiere el uso de un enfoque de desarrollo ágil. En este caso, XP es la metodología que se ha adoptado, dada su flexibilidad ante el cambio de

requisitos, el manejo de los riesgos técnicos, el tamaño del equipo de desarrollo.

- Para lograr una mayor organización en el proceso de desarrollo se selecciona el Visual Paradigm como herramienta CASE, a partir de él se selecciona el lenguaje UML para el modelado del sistema y con el objetivo de mantener la soberanía tecnológica se usó como lenguaje de programación Java y como IDE el Netbeans.

CAPÍTULO 2. Análisis y diseño de la propuesta de solución

El presente capítulo contiene la modelación del sistema propuesto haciendo uso de los artefactos de la metodología seleccionada. También contiene las características de la metodología, de las herramientas de diseño y del lenguaje de programación; además contiene las características de los elementos necesarios para el proceso de análisis, diseño, implementación y prueba. Muestra las características del sistema, definidas a partir de los requisitos funcionales y no funcionales especificados en el capítulo con el fin de garantizar un correcto funcionamiento de la aplicación.

2.1. Solución propuesta

Haciendo uso de las distintas herramientas, tecnologías y marcos de trabajo actuales se propone implementar un sistema que contenga una serie de métodos con el fin de agilizar y facilitar el proceso de toma de decisiones bajo incertidumbre con múltiples criterios. El sistema cumplirá con una serie de requisitos funcionales y no funcionales que permitirán una mejor interacción del decisor con el mismo, haciendo su trabajo más eficiente y garantizando que la toma de decisiones sea correcta y precisa. El sistema parte de la conformación del problema, donde el usuario tiene que elegir la norma ISO deseada, esta contiene el conjunto de criterios a tener en cuenta, continuando con la inserción de las alternativas, desde ahí, a través del editor de problema el usuario podrá hacer modificaciones a los mismos en caso de que así lo desee. A continuación, el usuario podrá elegir el método a usar, el cual le brindará la propuesta de solución final.

La herramienta incluye además un módulo de comparación donde después de haber realizado el proceso con distintos métodos permite compararlos viendo semejanzas y diferencias entre sus resultados, permitiendo así llegar a una mejor solución.

La siguiente figura muestra un esquema de la solución propuesta.

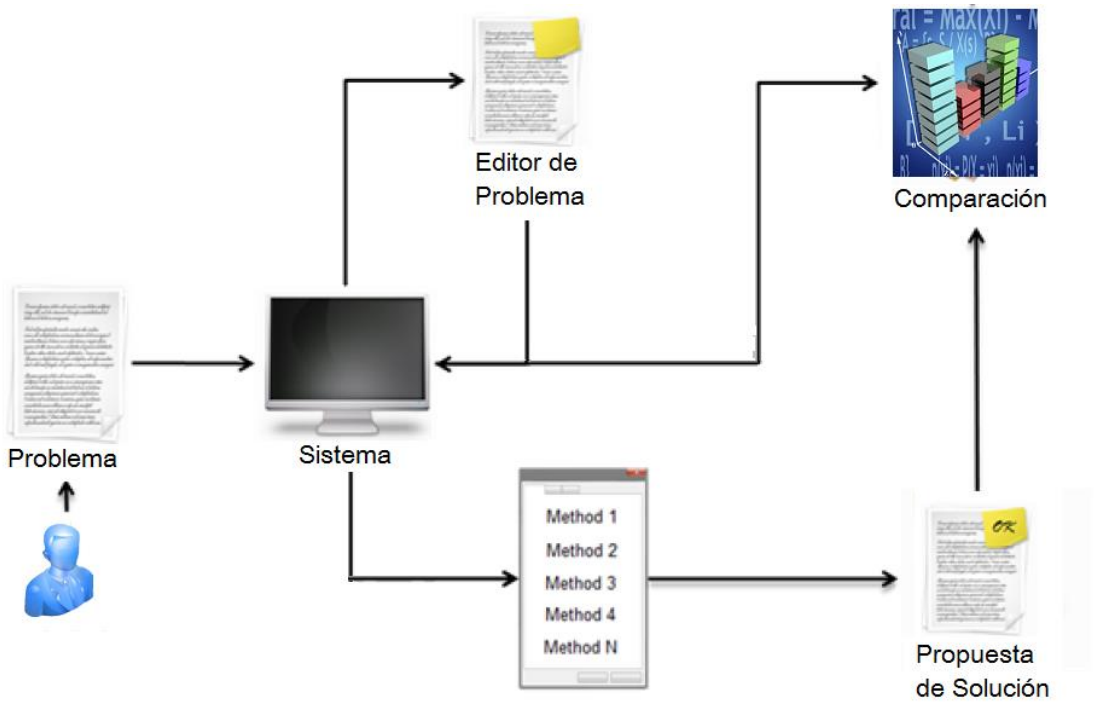


Figura 2.1: Descripción de la solución Propuesta. Elaboración propia.

2.2. Exploración. Requisitos del software

Los requisitos de software se pueden clasificar en dos tipos, funcionales y no funcionales. Los requisitos funcionales son aquellas condiciones y capacidades que debe cumplir el sistema sin afectar su correcto funcionamiento. Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema, pueden ser restricciones de tiempo, de los estándares a utilizar, de rendimiento, entre otros (44).

2.2.1. Historia de Usuario

Las Historias de Usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software y constituyen el artefacto más importante de esta metodología. Se tratan de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden desecharse, remplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU debe ser comprensible y delimitada para que los programadores puedan implementarlas en unas semanas (42).

En el presente trabajo se obtuvieron un total de 17 HU, cada una de ellas respondiendo a las diferentes funcionalidades solicitadas por el cliente, las cuales se realizarán en 4 iteraciones. Teniendo en cuenta la dimensión de la memoria escrita, se realiza una selección de las HU de mayor importancia. A continuación, se describen 4 de estas, otras 13 se describen en el Anexo I.

Tabla 2. Historia de usuario “Adicionar Criterios”

Historia de usuario	
Número: 1	**Nombre:** Adicionar Criterios
Responsable: David Pino González	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 8 horas
Complejidad de desarrollo: Baja	
Descripción: El sistema permite crear y añadir un nuevo criterio en el sistema.	

Tabla 3.Historia de usuario “Adicionar Alternativa”

Historia de usuario	
Número: 2	**Nombre:** Adicionar Alternativa
Responsable: David Pino González	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 8 horas
Complejidad de desarrollo: Baja	
Descripción: El sistema permite crear y añadir un nuevo software al conjunto de alternativas.	

Tabla 4.Historia de usuario “Importar Testing de expertos”

Historia de usuario	
Número: 3	**Nombre:** Importar Testing de expertos
Responsable: Tony Castillo Martin	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 20 horas
Complejidad de desarrollo: Media	
Descripción: El sistema carga los datos a partir de un Excel y los muestra en una vista, permitiendo modificar sus valores. Entrada: Matriz compuesta por valores numéricos. Salida: Visualización de la matriz de en una vista.	

Tabla 5.Historia de usuario “Importar Normas ISO enfocadas en parámetros de calidad”

Historia de usuario	
Número: 4	**Nombre:** Importar Normas ISO enfocadas en parámetros de calidad.
Responsable: Tony Castillo Martin	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 20 horas
Complejidad de desarrollo: Baja	
Descripción: El sistema carga los datos desde una base de datos y los muestra en una vista, permitiendo modificar sus valores. Entrada: Parámetros o criterios de la norma con sus propiedades. Salida: Visualización de la criterios y propiedades.	

El tiempo estimado para cada Historia de Usuario está dado en horas, donde 8 horas de trabajo representa la cantidad de horas laborables por día.

2.2.2. Requisitos No Funcionales

Se tuvieron en cuenta 7 requisitos no funcionales, cada uno de ellos se encuentra explicado en la siguiente tabla:

Tabla 6. Requisitos no funcionales.

Requisitos No Funcionales:		
1	**Requerimientos de software:** El sistema podrá ser ejecutado desde sistemas operativos Linux, Windows 7/8 y versiones superiores. Para un correcto funcionamiento la versión de la Máquina Virtual de Java requerida es 7u21 o superiores, y se utilizará para la persistencia de los datos PostgreSQL en su versión 9.1 o superior.	Analista
2	**Apariencia o interfaz externa:** La interfaz tendrá los componentes visuales necesarios para las operaciones del usuario sin sobrecarga de imágenes.	Analista
3	**Confiabilidad:** Comprobar que los datos proporcionados por los usuarios sean valores válidos. Garantizar que el sistema esté disponible las 24 horas del día para garantizar el acceso en todo momento.	Analista
4	**Portabilidad:** Permitir que la aplicación se ejecute en Sistemas Operativos como Linux y Windows, garantizando un producto multiplataforma.	Analista
5	**Requerimiento de Hardware:** La computadora que hará uso del sistema a	Analista

	desarrollar tendrá como velocidad de la CPU (*Central Processing Unit)* a 2.2 GHZ como mínimo. La memoria RAM (*Random-Access Memory*) debe tener como mínimo 2 Gb o superior a este.	
6	**Usabilidad:** La solución debe tener una interfaz gráfica atractiva permitiendo el uso del sistema sin necesidad de mucho conocimiento informático y el acceso a todas sus funcionalidades de manera sencilla y directa.	Analista
7	**Restricciones de diseño:** La aplicación se desarrollará en Java como lenguaje de programación, Visual Paradigm será la herramienta CASE a utilizar, se hará uso de Netbeans 7.3 como herramienta IDE y se utilizará como gestor de bases de datos PostgreSQL.	Analista

2.3. Planificación

2.3.1. Plan de Iteraciones

El plan de iteraciones es uno de los artefactos propuestos por la metodología XP para organizar el proceso de desarrollo del sistema, mostrando la duración y el orden en que serán implementadas las HU de la aplicación en cada iteración.

Tabla 7.Plan de iteraciones

Iteración	Historia de usuario	Duración estimada (semanas)
1	Adicionar Criterios	2 semanas y 1 día
	Modificar Criterios	
	Eliminar Criterios	
	Adicionar Alternativa	
	Modificar Alternativas	
	Eliminar Alternativas	
	Importar Testing de expertos	
	Importar Normas ISO enfocadas en parámetros de calidad	
2	Ponderar los criterios según la importancia	5 semanas
	Modificar los pesos de los criterios según el método seleccionado	
	Elegir la mejor alternativa según Método Suma Ponderada	
3	Elegir la mejor alternativa según el Método Estructura Plana	8 semanas
	Elegir la mejor alternativa según Método AHP	
	Elegir la mejor alternativa según Método TOPSIS	

4	Elegir la mejor alternativa según Método Versus	7 semanas
	Obtener ranking de alternativas	
	Comparar resultados de los métodos entre sí	

2.4. Iteraciones

2.4.1. Tarjetas CRC

Las tarjetas CRC (Clase-Responsabilidad-Colaborador) es un artefacto generado por la metodología XP para diseñar la solución informática según el paradigma de la programación orientada a objetos (42).

En el proceso de diseño las tarjetas CRC jugaron un importante papel debido a que sirvieron de base para el modelo entidad relación, se confeccionó a través de ellas las bases de datos del sistema. Cada tarjeta CRC se convirtió en un objeto, sus responsabilidades en métodos públicos y sus colaboradores en llamados a otras clases.

Tabla 8.Tarjeta CRC de la clase “AHP”

Clase: AHP	
Responsabilidades:	**Colaboradores:**
Lista de descendientes (Devuelve el árbol de descendencia de un criterio dado)	Controladora
Lista de inmediatos (Devuelve el listado de hijos inmediatos)	Controladora
Alternativa dado un Id (Devuelve la alternativa según el Id pasado por parámetro)	Alternativa
Criterio dado un Id (Devuelve el criterio según el Id pasado por parámetro)	Criterio
Alternativa dado una posición (Devuelve la alternativa según la posición pasada por parámetro)	Alternativa
Criterio dado una posición (Devuelve el criterio según la posición pasada por parámetro)	Criterio
Iniciar (Recibe las matrices con los valores para la operación)	MatrizResult
Finalizar (termina modificando los pesos de las alternativas a su estado final ejecutando el modelo matemático del método AHP)	

Tabla 9.Tarjeta CRC de la clase “Suma Ponderada”

Clase: Suma Ponderada	
Responsabilidades:	**Colaboradores:**
Lista de descendientes (Devuelve el árbol de descendencia de un criterio dado)	Controladora
Padres (Devuelve el listado criterios del nivel 0)	Controladora
Rellena (Calcula el peso de los criterios hojas haciendo un recorrido a lo ancho, teniendo este la lógica del método)	Controladora

Clase: **Suma Ponderada**	
Responsabilidades:	**Colaboradores:**
Criterio dado un Id (Devuelve el criterio según el Id pasado por parámetro)	Criterio
Criterio dado una posición (Devuelve el criterio según la posición pasada por parámetro)	Criterio
Alternativa dado un Id (Devuelve la alternativa según el Id pasado por parámetro)	Alternativa
Alternativa dado una posición (Devuelve la alternativa según la posición pasada por parámetro)	Alternativa

2.5. Arquitectura de software

La arquitectura de software es el conjunto de técnicas metodológicas desarrolladas con el fin de facilitar la programación. Hace referencia a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiarse en el desarrollo de software dentro de un sistema informático. Establece todos los fundamentos para que los analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema. (45)

2.5.1. Arquitectura N Capas

La programación por capas es una arquitectura cliente-servidor en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. También, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de los niveles, de forma que basta con conocer la API (*Application Programming Interface*) que existe entre niveles. (46)

Se seleccionó específicamente la arquitectura N capas para el desarrollo del sistema, con N igual a tres, ya que brinda la posibilidad de adaptarse a los requerimientos que debe cumplir el sistema, permite la reusabilidad y facilita la realización de las pruebas.

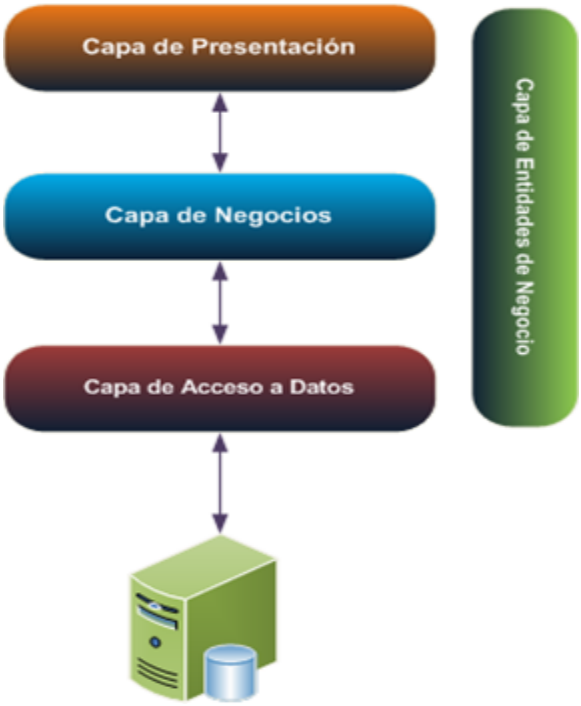


Figura 2.1: Arquitectura 3 capas. **(47)**

2.6. Patrones de diseño

Los patrones de diseño brindan un esquema que facilita el trabajo y aportan mayor organización y claridad en la estructura de la aplicación. El sistema que se desea implementar basa su arquitectura en 3 capas, empleando patrones arquitectónicos y de diseño, donde tiene gran importancia la utilización de patrones GRASP y los patrones GOF.

2.6.1. Patrones GRASP (Patrones de Software para la Asignación General de Responsabilidad)

"Los patrones **GRASP** describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable." (48)

El uso de estos patrones es un eslabón fundamental para el desarrollo de una aplicación con la calidad requerida. Estos patrones se describen a continuación:

Bajo acoplamiento: El Bajo Acoplamiento es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Estimula asignar una

responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. (48)

Alta Cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Representa una clase con responsabilidades moderadas en un área funcional, colaborando con otras para concretar tareas. Diseño más claro y comprensible. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. (48)

Experto: Es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. Ofrece una analogía con el mundo real. (48)

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento, se debe buscar una clase de objeto que agregue, contenga y realice otras operaciones sobre este tipo de instancias. (48)

Controlador: Es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos. (48)

2.6.2. Otros patrones importantes

Los patrones GOF se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro "*Design Patterns—Elements of Reusable Software*" de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, a partir de entonces estos patrones son conocidos como los patrones de la pandilla de los cuatro (GoF, *gang of four*), de estos patrones se utilizaron los siguientes en el desarrollo del sistema:

Patrón DAO: El problema que viene a resolver este patrón es el de contar con diversas fuentes de datos (base de datos, archivos, servicios externos, etc.). De tal forma que se encapsula la forma de acceder a la fuente de datos. (49)

Patrón Fachada: Es un patrón de diseño de tipo estructural. Proporciona una interfaz unificada de alto nivel para un subsistema, que oculta las interfaces de bajo nivel de las clases que lo implementan. Con esto se consiguen dos objetivos fundamentales: hacer el subsistema más fácil de usar y desacoplar a los clientes de las clases del subsistema. (50)

A continuación, a través del diagrama de paquetes se explicará el desarrollo de la arquitectura y de los patrones en la aplicación.

2.7. Diagrama de Paquetes

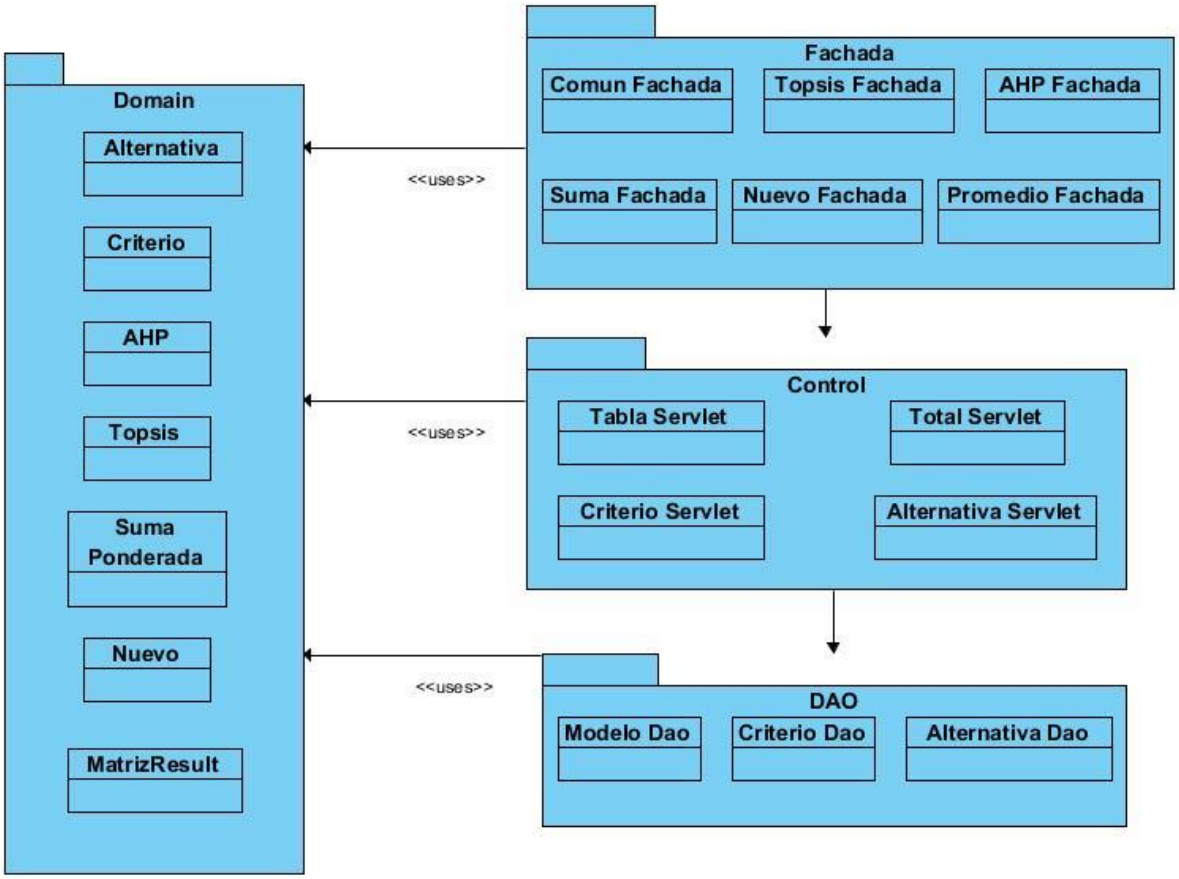


Figura 2.2: Diagrama de Paquetes.

En el diagrama de paquetes se muestra el uso de la arquitectura N Capas. El paquete Control contiene las clases principales de la aplicación, su función inicial es redireccionar a la interfaz de bienvenida, esta a su vez contiene una vista con los datos iniciales del sistema. Una vez dentro, el usuario podrá elegir entre las distintas interfaces, donde cada una responde a los requerimientos del sistema.

Haciendo uso de la arquitectura del patrón fachada se brinda una interfaz común entre las clases. Se podrá acceder al paquete Control donde se encuentra la lógica del sistema, y a su vez, a través de este se puede acceder al paquete donde se encuentran las clases encargadas del acceso a la base de datos, haciendo uso del patrón DAO. Este patrón permite que el software no tenga que manejar la ubicación donde se almacenan los datos, ni la manera en que se debe realizar esta actividad. Todos estos paquetes (DAO, Fachada, Control) hacen uso de las clases del paquete Dominio. Las clases que se encuentran en el paquete Control tienen asignada la responsabilidad de llevar a cabo todas las operaciones y cálculos matemáticos del método al que hacen referencia, haciendo uso de los datos entrados por el usuario además de mantener un mínimo de dependencia entre ellas, reflejando el uso de los patrones alta cohesión y bajo acoplamiento. El patrón experto se encuentra presente en las clases AHP, TOPSIS, Suma Ponderada, debido a que no se encuentran sobrecargadas de responsabilidades y los objetos de las mismas realizan sus actividades con la información que poseen. También en estas clases se encuentra el patrón creador ya que los objetos creados poseen las responsabilidades pertinentes asignadas por su creador. En conjunto con estos se encuentra el patrón controlador debido a la existencia de clases encargadas de asociar los eventos externos generados con las operaciones del sistema.

Consideraciones parciales.

A partir del análisis del sistema se arribó a las siguientes consideraciones:

- La definición de las 17 HU y los 7 requisitos no funcionales brinda una guía descriptiva de cómo se comportará la implementación del sistema.
- Los requisitos del sistema, la arquitectura de N capas y el uso de los patrones de diseño GRASP y GOF (Patrón DAO Patrón Fachada) permitieron obtener un diseño robusto de la aplicación.
- Las definiciones de las Tarjetas CRC aportaron en la identificación de las responsabilidades de las clases y las colaboraciones de las mismas.

CAPÍTULO 3. Implementación y Pruebas del Sistema

El presente capítulo tiene como objetivo esencial presentar los resultados de la implementación y pruebas realizadas a la aplicación. Se describen los estilos de programación utilizados, las pruebas unitarias realizadas al código y las pruebas de aceptación realizadas en conjunto con el cliente.

3.1. Tareas de Ingeniería

La Metodología XP define como tareas de ingeniería a todas las tareas que se realizan en cada iteración. A cada una de las tareas se le asigna un programador, el cual será responsable de ejecutarla.

En la siguiente tabla se muestra las tareas de ingeniería definidas para la realización de las historias de usuario. En el Anexo III se encuentran las especificaciones de las tareas.

Tabla 10.Tareas de ingeniería

Iteración	Historia de Usuario	Tareas
1	Adicionar Criterios	-Establecer los criterios medibles
	Modificar Criterios	
	Eliminar Criterios	
	Adicionar Alternativa	-Establecer los software opcionales
	Modificar Alternativa	
	Eliminar Alternativa	
	Importar Testing de expertos	-Cargar Testing
	Importar Normas ISO enfocadas en parámetros de calidad.	-Cargar Normas ISO
2	Ponderar los criterios según la importancia	-Establecer valores para los criterios
	Modificar los pesos de los criterios según el método seleccionado	-Editar los valores de los criterios
	Elegir la mejor alternativa según Método Suma Ponderada	-Seleccionar el método Suma Ponderada
3	Elegir la mejor alternativa según le	-Seleccionar el método Estructura plana

	Método Estructura Plana	
	Elegir la mejor alternativa según Método AHP	-Seleccionar el método AHP
	Elegir la mejor alternativa según Método TOPSIS	-Seleccionar el método TOPSIS
4	Elegir la mejor alternativa según Método Versus	-Seleccionar el método Versus
	Obtener ranking de alternativas	-Mostrar las alternativas ordenadas
	Comparar resultados de los métodos entre sí	-Mostrar resultados de diversos métodos

3.2. Estilos de programación

Los estilos de programación son reglas que se utilizan para estructurar el código fuente de una aplicación en desarrollo. En este epígrafe se describirán los estándares y reglas de programación utilizadas en la implementación del sistema.

Nombre de las clases: El estilo de capitalización utilizado para la notación de las clases será el UpperCamelCase, por lo que cada subpalabra que conforme el nombre de la clase comenzará con una letra mayúscula.

- Se definieron un grupo de reglas con el fin de lograr una mayor claridad en el código:
- Utilizar nombres descriptivos para los nombres de las estructuras, atributos y parámetros.
- Los comentarios deben estar en el mismo nivel del código.
- Se deberá realizar una sola declaración por línea con el fin de facilitar los comentarios.
- No debe haber espacios en blanco entre los nombres de los métodos y el paréntesis ‘(’ que abre su lista de parámetros.
- Cuando una expresión no entra en una línea, se rompe de acuerdo con estos principios:

Romper después de una coma.

Romper antes de un operador.

Ejemplo:

SumaPonderada sumaPonderada = new SumaPonderada(listado, ad.listadopadre(getConnection()), matriz, result);

3.3. Diagrama de despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo familiar. En la figura que se muestra a continuación se representa el Diagrama de Despliegue del sistema.

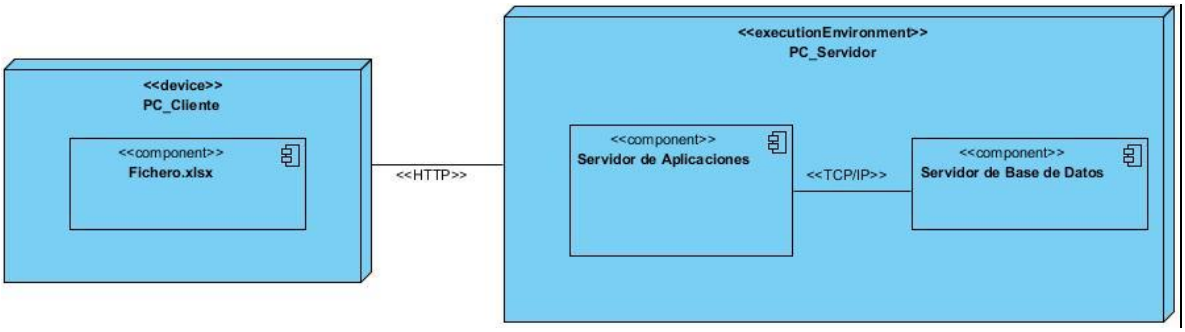


Figura 3.1: Diagrama de Despliegue del sistema.

3.4. Pruebas de software

En el proceso de desarrollo de software una de las fases más importante es la fase de pruebas, debido a que a través de ella se validan que los requisitos del software han sido cumplidos y garantiza que el sistema posee la calidad requerida.

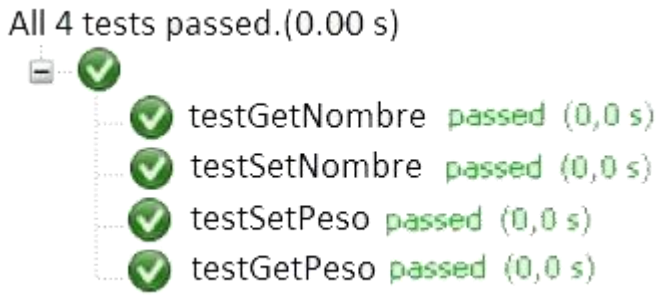
3.4.1. Pruebas Unitarias

Las pruebas unitarias son diseñadas por los desarrolladores con el objetivo de verificar el código y detectar posibles errores durante la ejecución del software a las funcionalidades de las clases.

Las pruebas unitarias se realizaron utilizando la librería JUnit del entorno de desarrollo integrado NetBeans. Esta librería permite realizar la ejecución de clases Java de manera automatizada y controlada para comprobar si el funcionamiento

de las funcionalidades de una clase se comporta de la manera esperada. A continuación, se muestra un ejemplo de prueba unitaria realizada al sistema, el resto de las pruebas se encuentran en el Anexo IV.

Tabla 11.Prueba unitaria para la clase “MatrizResult”

Prueba unitaria	
Nombre: testMatrizResult	
Estado: Satisfactorio	**Tipo de prueba:** Caja blanca
Ejecutado por: Tony Castillo Martin	**Verificado por:** Luis Manuel Valera Pérez
Criterio de aceptación: Obtener un ranking ordenado de alternativas	
Resultado: 	

Análisis de las pruebas unitarias

Luego de haber implementado las historias de usuario planificadas en la primera iteración del desarrollo del sistema se realizó la primera iteración de pruebas unitarias detectándose veinticinco no conformidades (NC). La mayor parte de estas, estaban asociadas a la entrada de datos al sistema y al manejo del árbol de criterios en cada método. Luego de haber implementado las historias de usuario planificadas en la segunda iteración del desarrollo del sistema, corregidas ya todas las NC encontradas, se realizó la segunda iteración de pruebas unitarias detectándose catorce NC, dos de las cuales estaban asociadas al manejo del árbol de decisiones y a la visualización de los datos de salidas que muestra el sistema, el resto a la solución del problema. Luego de haber implementado las historias de usuario planificadas en la tercera iteración del desarrollo del sistema y corregidas todas las NC encontradas anteriormente, se realizó una tercera

iteración de pruebas unitarias detectándose tres NC. Luego de ser corregidas las NC detectadas en la tercera iteración, se realizó una cuarta iteración de pruebas unitarias en la que no se detectaron NC obteniendo un resultado satisfactorio, realizándose un total de cuatro iteraciones de pruebas unitarias. El resultado de las pruebas unitarias en cada iteración se puede apreciar en la gráfica que se muestra a continuación.

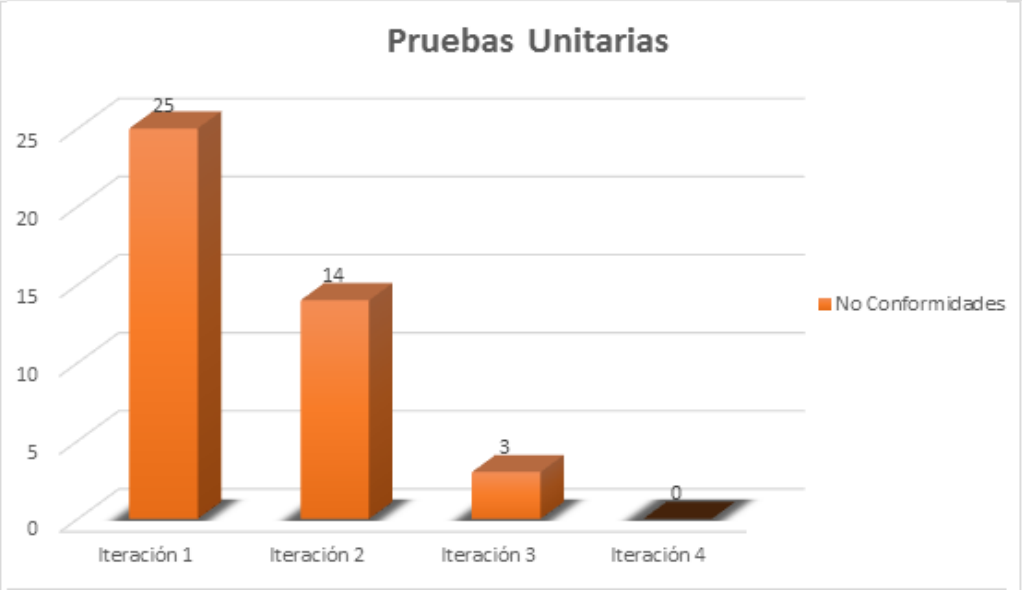


Figura 3.2: Gráfica de no conformidades detectadas en las pruebas unitarias.

3.4.2. Pruebas de Aceptación

Las pruebas de aceptación tienen como objetivo fundamental validar el nivel la satisfacción del cliente con respecto al software desarrollado. A continuación, se muestra un ejemplo de prueba de aceptación realizada al sistema, en el Anexo V podrán encontrar el resto de las pruebas de aceptación realizadas.

Tabla 12.Prueba de aceptación de la historia de usuario “Importar Testing de expertos”

Prueba de aceptación	
Nombre: Prueba de aceptación Importar Testing	**Historia de usuario:** Importar Testing de expertos
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para importar el Test de los expertos al sistema.	
Condiciones de ejecución: Debe haberse elegido un método y una norma ISO.	
Entrada/Pasos de ejecución: - El usuario selecciona el botón pruebas. - Selecciona en la interfaz el botón browser. - Se busca la ubicación del archivo que contiene los datos. - Luego el usuario selecciona el botón aceptar.	
Resultado esperado: Se cargan los datos en la tabla perteneciente a la interfaz	
Evaluación de la prueba: Prueba satisfactoria	

Análisis de las pruebas de aceptación

En las pruebas de aceptación se realizaron cinco iteraciones. Una vez implementadas las historias de usuario de la primera iteración de la implementación del sistema se realizó la primera iteración de las pruebas de aceptación, se detectó en la misma once no conformidades, de estas cinco estaban relacionadas con la entrada de datos al sistema y las restantes en la devolución de los resultados en el área de trabajo.

Después, una vez implementadas las historias de usuario de la segunda iteración y haber corregido las no conformidades detectadas en la iteración anterior, se desarrolló la segunda iteración de pruebas de aceptación detectándose cuatro no conformidades relacionadas con el correcto funcionamiento del método Suma Ponderada. Luego de haber sido corregidas las no conformidades de la segunda iteración de prueba y estar implementadas las historias de usuario de la tercera iteración se realizó la tercera iteración de las pruebas de aceptación, detectándose 6 no conformidades relacionadas a la devolución de los resultados al hacer uso de los métodos AHP y TOPSIS. Luego de ser corregidas las no conformidades de la tercera iteración y estar desarrolladas las historias de usuario de la cuarta iteración, se pasó a realizar la cuarta iteración de pruebas de aceptación

detectándose tres no conformidades relacionadas con el con la comparación de los de los resultados arrojado por los métodos utilizados. Una vez corregidas las no conformidades de la cuarta iteración de pruebas de aceptación se realizó la quinta y última iteración de prueba de aceptación no detectándose así ninguna no conformidad obteniéndose un resultado de satisfactorio. A continuación, se muestra una gráfica que contiene los resultados de las pruebas de aceptación.

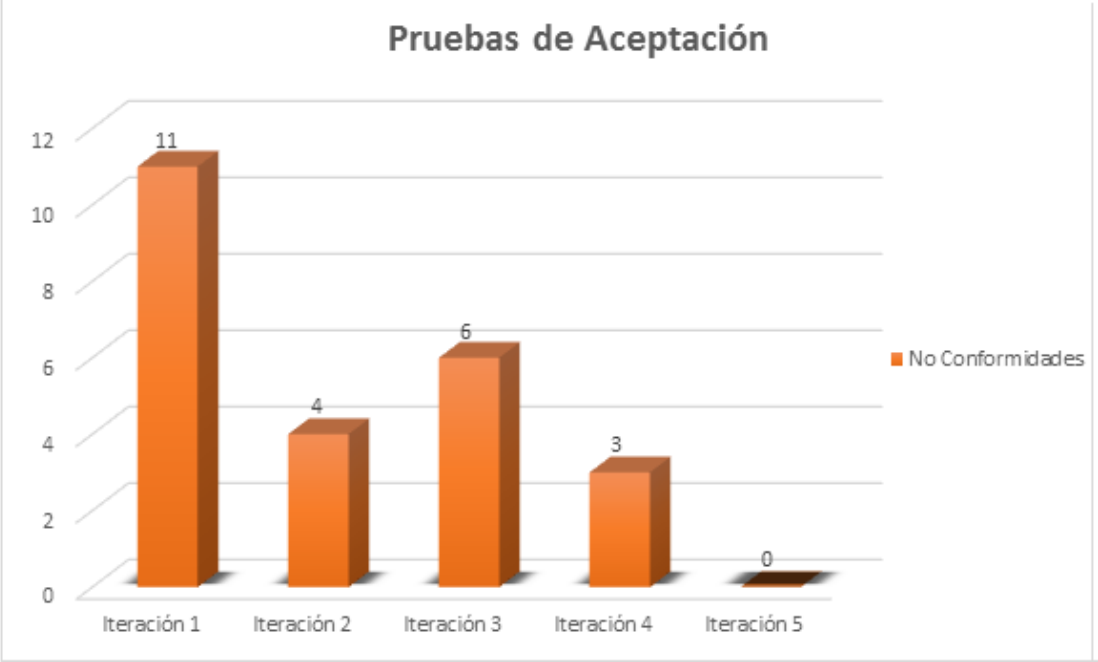


Figura 3.3: Gráfica de no conformidades detectadas en las pruebas de aceptación.

Consideraciones parciales

En el presente capítulo se describen los elementos necesarios para completar el proceso de implementación y prueba del sistema SofQualityAssess. Se describen las Tareas de Ingeniería, así como los estilos de programación utilizados en la implementación del sistema. También se realiza el análisis de los resultados arrojados por las pruebas de software realizadas a la aplicación llegando a las siguientes consideraciones:

- Con la definición de las 13 tareas de ingeniería se evitó la sobrecarga de trabajo en el equipo de desarrollo y se agilizó el proceso.
- La aplicación de las pruebas unitarias permitió validar las funcionalidades críticas del sistema, mientras que las pruebas de aceptación en conjunto con el cliente, posibilitaron comprobar la correcta implementación de las historias de usuarios definidas con anterioridad.
- Con la realización de las pruebas de software se llegó a la conclusión que la implementación satisface los requerimientos definidos por el cliente.
- La solución propuesta brinda una mejoría en el proceso de toma de decisiones haciendo el mismo más eficiente y eficaz debido a que fueron informatizados todos los cálculos y los resultados de calidad son semejantes a los que reflejan los productos de software en explotación.

CONCLUSIONES

A partir del desarrollo de la investigación presente se logra arribar a las siguientes conclusiones generales:

1. A partir del estudio realizado, se demostró que el proceso de evaluación de la calidad de los productos software se comporta como un proceso de toma de decisión multicriterio.
2. El uso de un enfoque de desarrollo ágil con XP, así como de herramientas y tecnologías tales como Visual Paradigm, UML, el lenguaje de programación Java y Netbeans facilitó el correcto desarrollo del sistema informático.
3. Los requisitos del sistema, la arquitectura de N capas y el uso de los patrones de diseño permitieron obtener un diseño robusto, flexible y óptimo de la aplicación.
4. Con la realización de las pruebas de software se demostró que la implementación satisface los requisitos definidos por el cliente.
5. La solución propuesta hace más eficiente y eficaz el proceso de toma de decisión en la evaluación de la calidad de los productos de software. Además, brinda beneficios al cliente tales como la incorporación de nuevos métodos de solución, así como un módulo de comparación de los resultados arrojados por diferentes métodos de agregación de la información.

RECOMENDACIONES

Teniendo en cuenta los resultados obtenidos en la presente investigación y basado en la experiencia adquirida, se recomienda:

1. Realizar el tratamiento de la incertidumbre fundamentado en los conceptos de la teoría de conjuntos difusos.
2. Extender el uso de la aplicación a otros centros que realizan la evaluación de la calidad de software.

Referencias Bibliográficas

1. Rodriguez Gomez, Jorge Luis. Ingenieria de Software: Calidad de Software. [En línea] [Citado el: 23 de enero de 2016.] http://ingenieriasw2.blogspot.com/p/calidad-de-software.html.

2. Fernández Carrasco, Oscar M., García León, Delba y Beltrán Benavides, Alfa . Un enfoque actual sobre la calidad del software.

3. Millet Lombida, Yanetsi. *Procedimiento para incorporar la Ingeniería de Usabilidad en el proceso de desarrollo del software.* Universidad de las Ciencias Infirmáticas. La Habana : s.n., 2015. Tesis de maestria.

4. Singh y K. Dubey, S. Evaluation of Usability Using Soft Computing Technique. *Int. J. Sci. Eng. Res.* 2013, Vol. 4, 12, págs. 162–166.

5. Morisio, M. y Tsoukias, A. IusWare: a methodology for the evaluation and selection of software products. *Softw. Eng. IEE Proc.* 1997, Vol. 144, 3, págs. 162–174.

6. Stamelos, I. , y otros. Knowledge based evaluation of software systems : a case study. *Inf. Softw. Technol.* 2000, Vol. 42, págs. 333–345.

7. Challa, J. S. , y otros. Integrated Software Quality Evaluation : A Fuzzy Multi-Criteria Approach. *J. Inf. Process. Syst.* 2011, Vol. 7, 3, págs. 473–518.

8. KNOW-HOW - Enciclopedia de Economía. [En línea] [Citado el: 23 de enero de 2016.] http://www.economia48.com/spa/d/know-how/know-how.htm. .

9. *Sitio oficial de la ISO.* [En línea] [Citado el: 7 de febrero de 2016.] http://www.iso.org/iso/home.htm..

10. Gestión de la calidad del software – Norma ISO 9126. *gestion de la calidad del software - norma iso-9126.* [En línea] 6 de junio de 2015. [Citado el: 22 de enero de 2016.]

11. Caponi, Marcelo, y otros. *Evaluación de Productos.* Universidad de la Republica –Facultad de Ingeniería.

12. Al-Kilidar, H., Cox, K. y Kitchenham, B. *The use and usefulness of the ISO/IEC 9126 quality standard.* s.l. : International Symposium on Empirical Software Engineering, 2005, págs. 126–132.

13. INEN. *Ingeniería de Software – Requerimientos y Evaluación de Calidad del Producto de Software (square) – Modelo de Referencia y Guía de Medición (ISO/IEC 25020:2007, IDT).* Ecuador : s.n., 2014.

14. ISO/IEC 25020:2007- Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Measurement reference model and guide. [En línea] [Citado el: 22 de enero de 2016.] http://www.iso.org/iso/catalogue_detail.htm?csnumber=35744..

15. ISO. Normas ISO/IEC 25000. [En línea] [Citado el: 22 de enero de 2016.] http://iso25000.com/index.php/normas-iso-25000. 1.

16. Saaty, Thomas L. *Relative Measurement and its Generalization in Decision Making: Why Pairwise Comparisons are Central in Mathematics for the Measurement of Intangible Factors - The Analytic Hierarchy/Network Process.* 2008.

17. Peña Abreu, Marieta. *Modelo para análisis de factibilidad.* La Habana : s.n., 2012.

18. Munier, N. *Procedimiento fundamentado en la programación lineal para la selección de alternativas en proyectos de naturaleza compleja y con objetivos múltiples.* Valencia : s.n., 2011.

19. Ayub, Mohammed, Kabir, Md. Jonaed y Rabiul Alam, Md. Golam. *Personnel Selection Method Using Analytic Network Process (ANP) and Fuzzy Concept.* 2009.

20. Hwang, C.L. y Yoon, K. *Multiple attribute decision methods an applications.* Berlin : Springer, 1981.

21. A. Krohling, Renato y C. Campanharo, Vinicius. Fuzzy TOPSIS for group decision making: A case study for accidents with oil spill in the sea. *Elsevier.* 2010.

22. Jahanshahloo, G.R., Hosseinzadeh Lotfi, F. y Davoodi, A.R. *Extension of TOPSIS for decision-making problems with interval data: Interval efficiency.* 2009.

23. Sánchez Nievares, Adrián, Luis Díaz, Heidy y Berrillo Borrero, Sael José. *Aplicación al proceso de selección de personal del método TOPSIS difuso utilizando umbrales de veto y voto mayoritario.* La Habana : s.n., 2012.

24. Leyva Vázquez, Maikel Y., y otros. Mapas cognitivos difusos para la selección de proyectos de tecnologías de la información. *Contaduría y Administración.* 2013, Vol. 58, 4, págs. 95-117.

25. Pérez Rodríguez , Fernando y Rojo Alboreca, Alberto. Minerva. Repositorio Institucional da Universidade de Santiago de Compostela: MPC 2.0, software para la aplicación del método AHP de toma de decisiones multicriterio. [En línea] 23 de marzo de 2012. [Citado el: 6 de febrero de 2016.] http://hdl.handle.net/10347/3830.

26. McGinley, P. Decision analysis software survey. *ORMS Today.* [En línea] 2012. [Citado el: 11 de Noviembre de 2014.] http://www.orms-today.org/surveys/das/das.html.

27. Creative Decisions Foundation. Printable version of Help | Super Decisions. [En línea] [Citado el: 6 de febrero de 2016.] http://www.superdecisions.com/printable-version-of-help/..

28. Peña Táramo, Yanet. Implementación del método TOPSIS en un sistema de ayuda en el proceso de Toma de Decisiones Multicriterio. 2013.

29. Torres Estol, Reinier y Padrón Del Pico, Javier Enrique. Implementación del método promethee en un sistema de ayuda a la toma de decisión multicriterio. 2013.

30. Sánchez González, Idel Jorge y González Landeiro, Adrián Román. Implementación de los métodos Proceso de Análisis Jerárquico y Proceso Analítico en Red en un sistema de soporte a la toma de decisiones multicriterio. 2013.

31. SL, Media Ingea. Decision Making Helper (Windows). *Uptodown.com.* [En línea] 29 de mayo de 2014. [Citado el: 5 de febrero de 2016.] http://decision-making-helper.uptodown.com/windows..

32. SL. Media Ingea. WinQSB (Windows). *Uptodown.com.* [En línea] 29 de marzo de 2011. [Citado el: 4 de febrero de 2016.] http://decision-making-helper.uptodown.com/windows.

33. calendamaia. NetBeans. [En línea] Genbeta Dev, 09 de ene de 2014. [Citado el: 26 de enero de 2016.] http://www.genbetadev.com/herramientas/netbeans-1.

34. S. Pressman, Roger. *Ingenieria de software. un enfoque práctico (pressman 5th ed).* Quinta. s.l. : McGraw-Hill.

35. pgAdmin: PostgreSQL administration and management tools. [En línea] [Citado el: 26 de enero de 2016.] https://www.pgadmin.org/index.php.

36. ¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML. [En línea] [Citado el: 27 de enero de 2016.] http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163. .

37. U Pérez, Felipe. Lenguaje de Programación JavaScript. [En línea] [Citado el: 27 de enero de 2016.] http://www.larevistainformatica.com/JavaScript.htm. .

38. Lenguajes de programación, programación Java. [En línea] [Citado el: 26 de enero de 2016.] http://www.lenguajes-de-programacion.com/programacion-java.shtml..

39. Hipertextual. Qué es HTML5. [En línea] 28 de Mayo de 2013. [Citado el: 27 de enero de 2016.] http://hipertextual.com/archivo/2013/05/entendiendo-html5-guia-para-principiantes/. .

40. ¿Qué es SQL o lenguaje de consultas estructuradas? [En línea] [Citado el: 1 de febrero de 2016.] http://searchdatacenter.techtarget.com/es/definicion/SQL-o-lenguaje-de-consultas-estructuradas..

41. Barzanallana, Rafael. Apuntes. Ingeniería del software. Sistemas Informáticos. Nivel de madurez software. Informática Aplicada a la Gestión Pública. [En línea] Universidad de Murcia. [Citado el: 25 de enero de 2016.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html..

42. Cuccaro Goggi, Lucila Ana. *Adecuación de la metodología de desarrollo Extreme Programming a proyectos llevados a cabo en la materia Laboratorio III de la Facultad de Ingeniería de la Universidad Austral.* 2010.

43. Gálvez Alcalde, Nils, Gonzales Horna, CHristian y Tirado Torres, Jordy. *Metodologís Ágile para el Desarrollo de Software: Programación Extrema XP (Extreme Programing).* Facultad de Ciencias Físicas y Matemáticas, Escuela Profesional de Informática, Universidad Nacional de Trujillo Sede Valle Jequetepeque.

44. Sommerville, Ian. *Ingeniería del Software.* Madrid : Pearson Educacion.S.A, 2005. 84-7829-074-5.

45. Fernández Lanvin, Daniel. *Definición de una arquitectura de software para el diseño de aplicaciones web.* Oviedo : s.n., 2009.

46. Lopez, Eliazar. Arquitectura de N capas. *ACADEMIA.* [En línea] [Citado el: 7 de febrero de 2016.] http://www.academia.edu/10102692/Arquitectura_de_n_capas..

47. Cuéllar, Jose. Estilos arquitecturales básicos. [En línea] [Citado el: 3 de febrero de 2016.] http://josecuellar.net/estilos-arquitecturales-en-el-diseno-de-un-sistema/..

48. Larman, Craig. *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* s.l. : PRENTICE HAL, 1999. 970-17-0261-1.

49. Gamma, Erich. *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995.

50. Quirós, Demetrio. Patrones de Diseño Resumen del libro Gamma y otros, por Demetrio Quirós. [En línea] 3 de abril de 2013. [Citado el: 2 de febrero de 2016.] http://wainu.ii.uned.es/grados/segundo/copy_of_TLPr/apuntes/resumen-de-patrones-de-diseno/view.

51. Actualidad | MINCOM. [En línea] 2012. [Citado el: 2 de 2 de 2016.] http://www.mincom.gob.cu.

52. G. Areces, A. y Marquez Diaz, Iskael. *“Diseño e implementación de las capas de negocio y acceso a datos de los módulos Planificación y Ejecución de Visitas Familiares.*

53. Karimi, A.R., y otros. *Using of the Fuzzy TOPSIS and Fuzzy AHP Methods for wastewater treatment process selection.* 2011.

Anexo I: Historias de usuario

Tabla 13.Historia de usuario “Eliminar Criterios”

Historia de usuario	
Número: 5	**Nombre:** Eliminar Criterios
Responsable: David Pino González	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 8 hora
Complejidad de desarrollo: Baja	
Descripción: El sistema permite eliminar uno o varios criterios en el sistema.	

Tabla 14.Historia de usuario “Modificar Criterios”

Historia de usuario	
Número: 6	**Nombre:** Modificar Criterios
Responsable: David Pino González	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 8 horas
Complejidad de desarrollo: Baja	
Descripción: El sistema permite modificar los criterios que han sido cargados en el sistema en el sistema.	

Tabla 15.Historia de usuario “Eliminar Alternativa”

Historia de usuario	
Número: 7	**Nombre:** Eliminar Alternativa
Responsable: David Pino González	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 8 horas
Complejidad de desarrollo: Baja	
Descripción: El sistema permite eliminar uno o varios software que conforman al conjunto de alternativas.	

Tabla 16.Historia de usuario “Modificar Alternativa”

Historia de usuario	
Número: 8	**Nombre:** Modificar Alternativa
Responsable: David Pino González	**Iteración:** 1
Prioridad en el negocio: Alta	**Tiempo estimado:** 8 horas
Complejidad de desarrollo: Baja	
Descripción: El sistema permite modificar una o varias alternativas.	

Tabla 17.Historia de usuario “Ponderar los criterios según la importancia”

Historia de usuario	
Número: 9	**Nombre:** Ponderar los criterios según la importancia
Responsable: Tony Castillo Martin	**Iteración:** 2
Prioridad en el negocio: Alta	**Tiempo estimado:** 40 horas
Complejidad de desarrollo: Media	
Descripción: El sistema permite dar valores a los criterios según su importancia y relación entre ellos. Entrada: Criterios sin ponderación. Salida: Criterios ponderados.	

Tabla 18.Historia de usuario “Modificar los pesos de los criterios según el método seleccionado”

Historia de usuario	
Número: 10	**Nombre:** Modificar los pesos de los criterios según el método seleccionado
Responsable: Tony Castillo Martin	**Iteración:** 2
Prioridad en el negocio: Alta	**Tiempo estimado:** 40 horas
Complejidad de desarrollo: Media	
Descripción: El sistema actualiza los valores siguiendo los pasos del algoritmo seleccionado. Entrada: Criterios ponderados por el usuario. Salida: Criterios ponderados por los algoritmos que indica el método seleccionado.	

Tabla 19.Historia de usuario “Elegir la mejor alternativa según Método Suma Ponderada”

Historia de usuario	
Número: 11	**Nombre:** Elegir la mejor alternativa según Método Suma Ponderada
Responsable: Tony Castillo Martin	**Iteración:** 2
Prioridad en el negocio: Alta	**Tiempo estimado:** 120 horas
Complejidad de desarrollo: Alta	
Descripción: El sistema a través del modelo matemático del Método Suma Ponderada da un listado de los criterios ordenados según la calidad designada por el mismo. Entrada: Criterios ponderados por los algoritmos que indica el método seleccionado, software y testing de expertos. Salida: Software ordenados según la calidad designada por el método.	

Tabla 20.Historia de usuario “Elegir la mejor alternativa según le Método Estructura Plana”

Historia de usuario	
Número: 12	**Nombre**: Elegir la mejor alternativa según le Método Estructura Plana
Responsable: Tony Castillo Martin	**Iteración:** 3
Prioridad en el negocio: Alta	**Tiempo estimado:** 80 horas
Complejidad de desarrollo: Alta	
Descripción: El sistema a través del modelo matemático del Método Estructura Plana da un listado de los criterios ordenados según la calidad designada por el mismo. Entrada: Criterios ponderados, software y testing de expertos. Salida: Software ordenados según la calidad designada por el método.	

Tabla 21.Historia de usuario “Elegir la mejor alternativa según Método AHP”

Historia de usuario	
Número: 13	**Nombre:** Elegir la mejor alternativa según Método AHP
Responsable: Tony Castillo Martin	**Iteración:** 3
Prioridad en el negocio: Alta	**Tiempo estimado:** 120 horas
Complejidad de desarrollo: Alta	
Descripción: El sistema a través del modelo matemático del método AHP da un listado de los criterios ordenados según la calidad designada por el mismo. Entrada: Criterios ponderados, software y testing de expertos. Salida: Software ordenados según la calidad designada por el método.	

Tabla 22.Historia de usuario “Elegir la mejor alternativa según Método TOPSIS”

Historia de usuario	
Número: 14	**Nombre:** Elegir la mejor alternativa según Método TOPSIS
Responsable: Tony Castillo Martin	**Iteración:** 3
Prioridad en el negocio: Alta	**Tiempo estimado:** 120 horas
Complejidad de desarrollo: Alta	
Descripción: El sistema a través del modelo matemático del Método TOPSIS da un listado de los criterios ordenados según la calidad designada por el mismo. Entrada: Criterios ponderados por los algoritmos que indica el método seleccionado, software y testing de expertos. Salida: Software ordenados según la calidad designada por el método.	

Tabla 23.Historia de usuario “Elegir la mejor alternativa según Método Versos”

Historia de usuario	
Número: 15	**Nombre:** Elegir la mejor alternativa según Método Versus
Responsable: Tony Castillo Martin	**Iteración:** 4
Prioridad en el negocio: Alta	**Tiempo estimado:** 120 horas
Complejidad de desarrollo: Alta	
Descripción: El sistema a través del modelo matemático del Método Versus da un listado de los criterios ordenados según la calidad designada por el mismo. Entrada: Criterios ponderados por los algoritmos que indica el método seleccionado, software y testing de expertos. Salida: Software ordenados según la calidad designada por el método.	

Tabla 24.Historia de usuario “Obtener ranking de alternativas”

Historia de usuario	
Número: 16	**Nombre:** Obtener ranking de alternativas
Responsable: Tony Castillo Martin	**Iteración:** 4
Prioridad en el negocio: Alta	**Tiempo estimado:** 80 horas
Complejidad de desarrollo: Media	
Descripción: El sistema muestra el resultado de cada método en una misma vista ordenados según la calidad asignada por cada uno. Entrada: Software ordenados según la calidad designada por cada método. Salida: Tabla comparativa de los software ordenados según su calidad según cada método en una misma vista.	

Tabla 25.Historia de usuario “Comparar resultados de los métodos entre sí”

Historia de usuario	
Número: 17	**Nombre:** Comparar resultados de los métodos entre sí
Responsable: Tony Castillo Martin	**Iteración:** 4
Prioridad en el negocio: Alta	**Tiempo estimado:** 80 horas
Complejidad de desarrollo: Media	
Descripción: El sistema muestra el resultado de cada método en una misma vista permitiendo compararlos entre sí, usando fórmulas de correlación. Entrada: Tabla comparativa de los software ordenados según su calidad según cada método en una misma vista. Salida: Tablas comparativa con valores de correlación entre los distintos métodos, comparando semejanza entre ellos.	

Anexo II: Tarjetas CRC

Tabla 26.Tarjeta CRC de la clase “TOPSIS”

Clase: TOPSIS	
Responsabilidades:	**Colaboradores:**
Lista de descendientes (Devuelve el árbol de descendencia de un criterio dado)	Controladora
Lista de inmediatos (Devuelve el listado de hijos inmediatos)	Controladora
Alternativa dado un Id (Devuelve la alternativa según el Id pasado por parámetro)	Alternativa
Criterio dado un Id (Devuelve el criterio según el Id pasado por parámetro)	Criterio
Alternativa dado una posición (Devuelve la alternativa según la posición pasada por parámetro)	Alternativa
Criterio dado una posición (Devuelve el criterio según la posición pasada por parámetro)	Criterio
Iniciar (Recibe las matrices con los valores para la operación)	MatrizResult
Finalizar(termina modificando los pesos de las alternativas a su estado final ejecutando el modelo matemático del método TOPSIS)	

Tabla 27.Tarjeta CRC de la clase “Criterio”

Clase: Criterio	
Responsabilidades:	**Colaboradores:**
Get Id Criterio (Permite el acceso a la clase Criterio)	
Set Id Criterio (Permite el acceso a la clase Criterio)	

Tabla 28.Tarjeta CRC de la clase “Alternativa”

Clase: Alternativa	
Responsabilidades:	**Colaboradores:**
Set Id Alternativa (Permite el acceso a la clase Alternativa)	
Set Nombre (Permite el acceso a la clase Alternativa)	

Anexo III: Especificaciones de las Tareas de Ingeniería

Tabla 29.Tarea de ingeniería “Establecer los criterios medibles”

Tarea de ingeniería	
Número: 1	**Historia de usuario:** Adicionar Criterios, Modificar Criterios, Eliminar Criterios
Nombre: Establecer los criterios medibles	
Responsable: David Pino González	**Tipo:** Desarrollo
Fecha de inicio: 18/4/2016	**Fecha de terminación:** 20/4/2016
Descripción: El usuario crea o carga los criterios a tener en cuenta para la evaluación de la calidad del software, permitiéndole el sistema eliminarlo o modificarlo una vez creado el criterio de calidad.	

Tabla 30.Tarea de ingeniería “Establecer los software opcionales”

Tarea de ingeniería	
Número: 2	**Historia de usuario:** Adicionar Alternativa, Modificar Alternativa, Eliminar Alternativa
Nombre: Establecer los software opcionales	
Responsable: David Pino González	**Tipo:** Desarrollo
Fecha de inicio: 20/4/2016	**Fecha de terminación:** 22/4/2016
Descripción: El usuario añade los software que serán evaluados en el proceso de evaluación de la calidad de software, permitiéndole el sistema eliminarlo o modificarlo según lo desee el mismo.	

Tabla 31.Tarea de ingeniería “Cargar Testing”

Tarea de ingeniería	
Número: 3	**Historia de usuario:** Importar Testing de expertos
Nombre: Cargar Testing	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 25/4/2016	**Fecha de terminación:** 27/4/2016
Descripción: El usuario carga los resultados del testing de expertos a partir de un fichero externo que contiene la información.	

Tabla 32.Tarea de ingeniería “Cargar Normas ISO”

Tarea de ingeniería	
Número: 4	**Historia de usuario:** Importar Normas ISO enfocadas en parámetros de calidad
Nombre: Cargar Normas ISO	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 27/4/2016	**Fecha de terminación:** 29/4/2016
Descripción: El usuario carga la norma ISO que posee los parámetros de calidad.	

Tabla 33.Tarea de ingeniería “Establecer valores para los criterios”

Tarea de ingeniería	
Número: 5	**Historia de usuario:** Ponderar los criterios según la importancia
Nombre: Establecer valores para los criterios	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 2/5/2016	**Fecha de terminación:** 6/5/2016
Descripción: El usuario establece los valores de cada criterio según el nivel de importancia.	

Tabla 34.Tarea de ingeniería “Editar los valores de los criterios”

Tarea de ingeniería	
Número: 6	**Historia de usuario:** Modificar los pesos de los criterios según el método seleccionado
Nombre: Editar los valores de los criterios	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 9/5/2016	**Fecha de terminación:** 13/5/2016
Descripción: El usuario puede modificar los valores de los criterios una vez definido el método a utilizar en la evaluación de la calidad de software.	

Tabla 35.Tarea de ingeniería “Seleccionar el método Suma Ponderada”

Tarea de ingeniería	
Número: 7	**Historia de usuario:** Elegir la mejor alternativa según Método Suma Ponderada
Nombre: Seleccionar el método Suma Ponderada	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 16/5/2016	**Fecha de terminación:** 10/6/2016
Descripción: El usuario puede utilizar el método de Suma Ponderada para realizar la evaluación de la calidad del software que conforman las alternativas.	

Tabla 36.Tarea de ingeniería “Seleccionar el método Estructura Plana”

Tarea de ingeniería	
Número: 8	**Historia de usuario:** Elegir la mejor alternativa según le Método Estructura Plana
Nombre: Seleccionar el método Estructura Plana	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 10/6/2016	**Fecha de terminación:** 10/6/2016
Descripción: El usuario puede utilizar el método de Estructura Plana para realizar la evaluación de la calidad de los software que conforman las alternativas.	

Tabla 37.Tarea de ingeniería “Seleccionar el método AHP”

Tarea de ingeniería	
Número: 9	**Historia de usuario:** Elegir la mejor alternativa según Método AHP
Nombre: Seleccionar el método AHP	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 10/6/2016	**Fecha de terminación:** 10/6/2016
Descripción: El usuario puede utilizar el método AHP para realizar la evaluación de la calidad de los software que conforman las alternativas.	

Tabla 38.Tarea de ingeniería “Seleccionar el método TOPSIS”

Tarea de ingeniería	
Número: 10	**Historia de usuario:** Elegir la mejor alternativa según Método TOPSIS
Nombre: Seleccionar el método TOPSIS	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 10/6/2016	**Fecha de terminación:** 10/6/2016
Descripción: El usuario puede utilizar el método TOPSIS para realizar la evaluación de la calidad de los software que conforman las alternativas.	

Tabla 39.Tarea de ingeniería “Seleccionar el método Estructura Plana”

Tarea de ingeniería	
Número: 11	**Historia de usuario:** Elegir la mejor alternativa según Método Versus
Nombre: Seleccionar el Método Versus	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 10/6/2016	**Fecha de terminación:** 10/6/2016
Descripción: El usuario puede utilizar el método Versus para realizar la evaluación de la calidad de los software que conforman las alternativas.	

Tabla 40.Tarea de ingeniería “Mostrar las alternativas ordenadas”

Tarea de ingeniería	
Número: 12	**Historia de usuario:** Obtener ranking de alternativas
Nombre: Mostrar las alternativas ordenadas	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 10/6/2016	**Fecha de terminación:** 10/6/2016
Descripción: El usuario observar el resultado de ejecutar el método seleccionado para realizar la evaluación de la calidad de los software de forma ordenada.	

Tabla 41.Tarea de ingeniería “Mostrar resultados de diversos métodos”

Tarea de ingeniería	
Número: 13	**Historia de usuario:** Comparar resultados de los métodos entre sí
Nombre: Mostrar resultados de diversos métodos	
Responsable: Tony Castillo Martin	**Tipo:** Desarrollo
Fecha de inicio: 10/6/2016	**Fecha de terminación:** 10/6/2016
Descripción: El usuario puede observar los diferentes resultados arrojados por la ejecución de varios métodos para realizar la evaluación de la calidad de los software que conforman las alternativas.	

Anexo IV: Pruebas Unitarias

Tabla 42.Prueba unitaria para la clase “Criterios”

Prueba unitaria	
Nombre: testCriterios	
Estado: Satisfactorio	**Tipo de prueba:** Caja blanca
Ejecutado por: Tony Castillo Martin	**Verificado por:** Luis Manuel Valera Pérez
Criterio de aceptación: Obtener los dato del criterio deseado	
Resultado: 	

Tabla 43.Prueba unitaria para la clase “Alternativas”

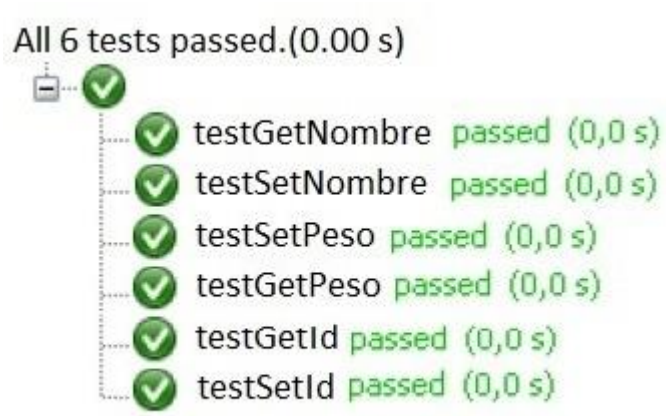
Prueba unitaria	
Nombre: testAlternativas	
Estado: Satisfactorio	**Tipo de prueba:** Caja blanca
Ejecutado por: Tony Castillo Martin	**Verificado por:** Luis Manuel Valera Pérez
Criterio de aceptación: Obtener los datos de la alternativa deseada	
Resultado: 	

Tabla 44.Prueba unitaria para el método “SumaPonderada”

Prueba unitaria	
Nombre: testSumaPonderada	
Estado: Satisfactorio	**Tipo de prueba:** Caja blanca
Ejecutado por: Tony Castillo Martin	**Verificado por:** Luis Manuel Valera Pérez
Criterio de aceptación: Obtener un ranking ordenado según la calidad de acuerdo al método Suma Ponderada	
Resultado: 	

Tabla 45.Prueba unitaria para el método “TOPSIS”

Prueba unitaria	
Nombre: testTOPSIS	
Estado: Satisfactorio	**Tipo de prueba:** Caja blanca
Ejecutado por: Tony Castillo Martin	**Verificado por:** Luis Manuel Valera Pérez
Criterio de aceptación: Obtener un ranking ordenado según la calidad de acuerdo al método TOPSIS	
Resultado: 	

Tabla 46.Prueba unitaria para el método “Versus”

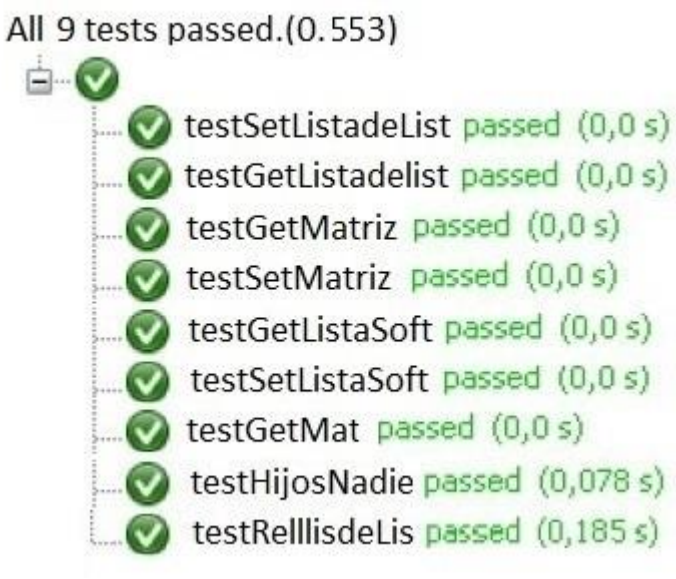
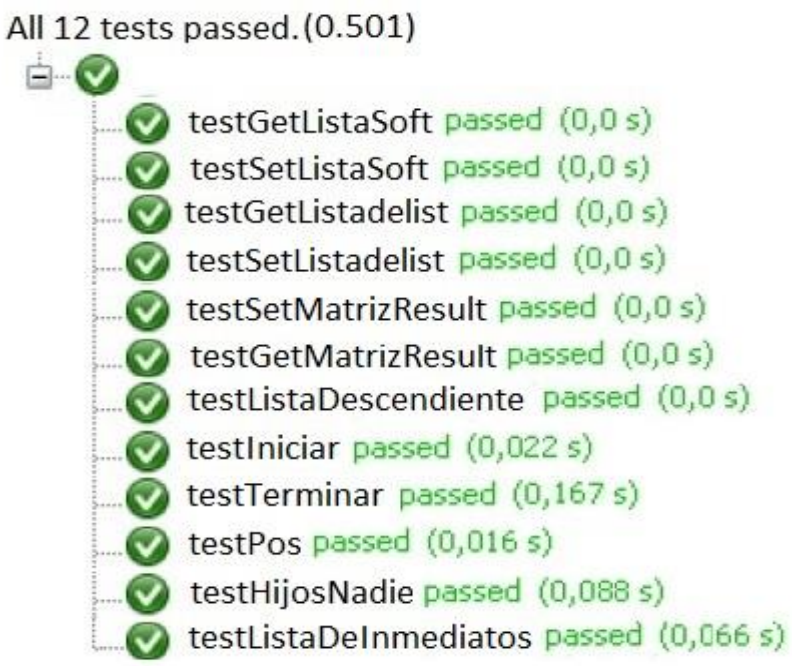
Prueba unitaria	
Nombre: testMetodoVersus	
Estado: Satisfactorio	**Tipo de prueba:** Caja blanca
Ejecutado por: Tony Castillo Martin	**Verificado por:** Luis Manuel Valera Pérez
Criterio de aceptación: Obtener un ranking ordenado según la calidad de acuerdo al método Versus	
Resultado: 	

Tabla 47. Prueba unitaria para el método “AHP”

Prueba unitaria	
Nombre: testAHP	
Estado: Satisfactorio	**Tipo de prueba:** Caja blanca
Ejecutado por: Tony Castillo Martin	**Verificado por:** Luis Manuel Valera Pérez
Criterio de aceptación: Obtener un ranking ordenado según la calidad de acuerdo al método AHP	
Resultado: 	

Anexo V: Pruebas de Aceptación

Tabla 48.Prueba de aceptación de la historia de usuario “Adicionar Criterios”

Prueba de aceptación	
Nombre: Prueba de aceptación gestionar criterios	**Historia de usuario:** Adicionar Criterios
Responsable: David Pino González	
Descripción: Prueba de funcionalidad para gestionar los datos de los criterios.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú la opción Criterios - Se muestra la interfaz perteneciente a dicha opción. - Se añade la información requerida. - Se selecciona el botón adicionar	
Resultado esperado: Se añade a la base de datos el criterio introducido por el usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 49.Prueba de aceptación de la historia de usuario “Modificar Criterios”

Prueba de aceptación	
Nombre: Prueba de aceptación gestionar criterios	**Historia de usuario:** Modificar Criterios
Responsable: David Pino González	
Descripción: Prueba de funcionalidad para gestionar los datos de los criterios.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú la opción Criterios - Se muestra la interfaz perteneciente a dicha opción. - Se selecciona el botón modificar del criterio deseado. - Se llenan los campos requeridos. - Se selecciona el botón aceptar.	
Resultado esperado: Se modifica en la base de datos el criterio introducido por el usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 50.Prueba de aceptación de la historia de usuario “Eliminar Criterios”

Prueba de aceptación	
Nombre: Prueba de aceptación gestionar criterios	**Historia de usuario:** Eliminar Criterios
Responsable: David Pino González	
Descripción: Prueba de funcionalidad para gestionar los datos de los criterios.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú la opción Criterios - Se muestra la interfaz perteneciente a dicha opción. - Se selecciona uno o varios criterios. - Se selecciona el botón eliminar	
Resultado esperado: Se elimina en la base de datos (el o los) criterio introducido por el usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 51.Prueba de aceptación de la historia de usuario “Adicionar Alternativa”

Prueba de aceptación	
Nombre: Prueba de aceptación gestionar alternativa	**Historia de usuario:** Adicionar Alternativa
Responsable: David Pino González	
Descripción: Prueba de funcionalidad para gestionar los datos de las alternativas.	
Condiciones de ejecución: El usuario debe tener ya definido sus posibles alternativas	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú la opción Alternativas. - Se muestra la interfaz perteneciente a dicha opción. - Se llenan los datos requeridos por el sistema. - Se selecciona el botón adicionar	
Resultado esperado: Se añade a la base de datos la alternativa introducida por el usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 52.Prueba de aceptación de la historia de usuario “Modificar Alternativa”

Prueba de aceptación	
Nombre: Prueba de aceptación gestionar alternativa	**Historia de usuario:** Modificar Alternativa
Responsable: David Pino González	
Descripción: Prueba de funcionalidad para gestionar los datos de las alternativas.	
Condiciones de ejecución: El usuario debe tener ya definido sus posibles alternativas	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú la opción Alternativas. - Se muestra la interfaz perteneciente a dicha opción. - Se selecciona el botón modificar de la alternativa a modificar. - Se llenan los parámetros requeridos. - Se selecciona la opción aceptar.	
Resultado esperado: Se modifica en la base de datos la alternativa introducida por el usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 53.Prueba de aceptación de la historia de usuario “Eliminar Alternativa”

Prueba de aceptación	
Nombre: Prueba de aceptación gestionar alternativa	**Historia de usuario:** Eliminar Alternativa
Responsable: David Pino González	
Descripción: Prueba de funcionalidad para gestionar los datos de las alternativas.	
Condiciones de ejecución: El usuario debe tener ya definido sus posibles alternativas	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú la opción Alternativas. - Se muestra la interfaz perteneciente a dicha opción. - Se seleccionan (la o las) alternativas a eliminar. - Se selecciona el botón eliminar. - Se confirma a eliminación seleccionando el botón aceptar	
Resultado esperado: Se añade a la base de datos la alternativa introducido por el usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 54.Prueba de aceptación de la historia de usuario “Importar Normas ISO enfocadas en parámetros de calidad”

Prueba de aceptación	
Nombre: Prueba de aceptación importar Normas ISO enfocadas en parámetros de calidad	**Historia de usuario:** Importar Normas ISO enfocadas en parámetros de calidad
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para importar las Normas ISO enfocadas en parámetros de calidad	
Condiciones de ejecución: Debe estar contenida la ISO en la base de datos.	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Inicio. - Selecciona la ISO con la cual realizar el proceso. - La ISO selecciona se carga directo desde la Base de Datos.	
Resultado esperado: Se carga la ISO seleccionada contenida en la base de datos.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 55.Prueba de aceptación de la historia de usuario “Ponderar los criterios según la importancia”

Prueba de aceptación	
Nombre: Prueba de aceptación ponderar los criterios según la importancia	**Historia de usuario:** Ponderar los criterios según la importancia
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para ponderar los criterios según la importancia de cada uno.	
Condiciones de ejecución: Debe existir el criterio en la base de datos	
Entrada/Pasos de ejecución: - El usuario selecciona la opción Criterios del menú principal. - Llena los datos requeridos, en este caso el peso del criterio. - Luego se añade a la base de datos seleccionando el botón adicionar.	
Resultado esperado: El criterio se encuentra almacenado en la base de datos con su respectivo peso.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 56.Prueba de aceptación de la historia de usuario “Modificar los pesos de los criterios según el método seleccionado”

Prueba de aceptación	
Nombre: Prueba de aceptación modificar los pesos de los criterios según el método seleccionado	**Historia de usuario:** Modificar los pesos de los criterios según el método seleccionado
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para modificar los pesos de los criterios según el método seleccionado.	
Condiciones de ejecución: Debe existir el criterio en la base de datos	
Entrada/Pasos de ejecución: - El usuario selecciona la opción Criterios contenida en el menú horizontal. - Se muestra la interfaz que posibilita modificar o eliminar los aspectos de los criterios. - Se selecciona el botón modificar y el criterio es modificado.	
Resultado esperado: Se modifica satisfactoriamente los criterios ya existentes.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 57.Prueba de aceptación de la historia de usuario “Elegir la mejor alternativa según Método Suma Ponderada”

Prueba de aceptación	
Nombre: Prueba de aceptación elegir la mejor alternativa según Método Suma Ponderada	**Historia de usuario:** Elegir la mejor alternativa según Método Suma Ponderada
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para elegir la mejor alternativa a través del Método Suma Ponderada	
Condiciones de ejecución: Debe encontrarse en base de datos las alternativas y lo criterios.	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Inicio. - Se selecciona como parámetro de entrada el método Suma Ponderada en la opción “Elije método a usar”. - Se selecciona el botón aceptar y se muestran los resultados en la interfaz de “Resultado”	
Resultado esperado: Se obtiene la correcta evaluación de calidad haciendo uso del método Suma Ponderada.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 58.Prueba de aceptación de la historia de usuario “Elegir la mejor alternativa según le Método Estructura Plana”

Prueba de aceptación	
Nombre: Prueba de aceptación elegir la mejor alternativa según le Método Estructura Plana	**Historia de usuario:** Elegir la mejor alternativa según le Método Estructura Plana
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para elegir la mejor alternativa a través del Método Estructura Plana	
Condiciones de ejecución: Debe encontrarse en base de datos las alternativas y lo criterios.	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Inicio. - Se selecciona como parámetro de entrada el método Estructura Plana en la opción “Elije método a usar”. - Se selecciona el botón aceptar y se muestran los resultados en la interfaz de “Resultado”.	
Resultado esperado: Se obtiene la correcta evaluación de calidad haciendo uso del método Estructura Plana.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 59.Prueba de aceptación de la historia de usuario “Elegir la mejor alternativa según Método AHP”

Prueba de aceptación	
Nombre: Prueba de aceptación elegir la mejor alternativa según Método AHP	**Historia de usuario:** Elegir la mejor alternativa según Método AHP
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para elegir la mejor alternativa según Método AHP	
Condiciones de ejecución: Debe encontrarse en base de datos las alternativas y lo criterios.	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Inicio. - Se selecciona como parámetro de entrada el método AHP en la opción “Elije método a usar”. - Se selecciona el botón aceptar y se muestran los resultados en la interfaz de “Resultado”	
Resultado esperado: Se obtiene la correcta evaluación de calidad haciendo uso del método AHP.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 60.Prueba de aceptación de la historia de usuario “Elegir la mejor alternativa según Método TOPSIS”

Prueba de aceptación	
Nombre: Prueba de aceptación elegir la mejor alternativa según Método TOPSIS	**Historia de usuario:** Elegir la mejor alternativa según Método TOPSIS
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para elegir la mejor alternativa a través del Método TOPSIS	
Condiciones de ejecución: Debe encontrarse en base de datos las alternativas y lo criterios.	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Inicio. - Se selecciona como parámetro de entrada el método TOPIS en la opción “Elije método a usar”. - Se selecciona el botón aceptar y se muestran los resultados en la interfaz de “Resultado”	
Resultado esperado: Se obtiene la correcta evaluación de calidad haciendo uso del método TOPSIS.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 61.Prueba de aceptación de la historia de usuario “Elegir la mejor alternativa según Método Versus”

Prueba de aceptación	
Nombre: Prueba de aceptación elegir la mejor alternativa según Método Versus	**Historia de usuario:** Elegir la mejor alternativa según Método Versus
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para elegir la mejor alternativa a través del Método Versus	
Condiciones de ejecución: Debe encontrarse en base de datos las alternativas y lo criterios.	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Inicio. - Se selecciona como parámetro de entrada el método Versus en la opción “Elije método a usar”. - Se selecciona el botón aceptar y se muestran los resultados en la interfaz de “Resultado”	
Resultado esperado: Se obtiene la correcta evaluación de calidad haciendo uso del método Versus.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 62.Prueba de aceptación de la historia de usuario “Obtener ranking de alternativas”

Prueba de aceptación	
Nombre: Prueba de aceptación obtener ranking de alternativas	**Historia de usuario:** Obtener ranking de alternativas
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para obtener el ranking de alternativas entradas al sistema	
Condiciones de ejecución: Debe haberse ejecutado un método de evaluación inicialmente	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Inicio. - Una vez ejecutado el método solución, seleccionar la opción Resultado, perteneciente a dicha interfaz.	
Resultado esperado: Se muestra satisfactoriamente el resultado arrojado por el método ejecutado de forma ordenada.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 63.Prueba de aceptación de la historia de usuario “Comparar resultados de los métodos entre sí”

Prueba de aceptación	
Nombre: Prueba de aceptación comparar resultados de los métodos entre sí	**Historia de usuario:** Comparar resultados de los métodos entre sí
Responsable: Tony Castillo Martin	
Descripción: Prueba de funcionalidad para comparar resultados arrojados por los métodos ejecutados	
Condiciones de ejecución: Debe haberse ejecutado al menos un método de evaluación.	
Entrada/Pasos de ejecución: - El usuario selecciona en el menú principal la opción Comparar. - Se muestran las funcionalidades pertenecientes a esta interfaz. - El usuario puede seleccionar los resultados de los métodos que desea mostrar en la comparación. - Se selecciona el botón mostrar y se muestran los elementos seleccionados por el usuario. - El usuario puede realizar la comparación entre los métodos de evaluación seleccionados, seleccionando el botón Correlación.	
Resultado esperado: Se muestran de los métodos ejecutados los seleccionados por el usuario.	
Evaluación de la prueba: Prueba satisfactoria	