Facultad 5

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

La priorización de requisitos basada en un proceso de computación con palabras

Autor: Alejandro Fuentes Barcelay

Tutor: Ing. Sailyn Salas Hechavarria

Ing. Luis Kayrumet Pérez Buigas

Co tutor: Ing. Manuel Alejandro Chang Fajardo

La Habana, Junio de 2015

"Año 57 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 3 días del mes de Julio del año 2015.

Autor:

_____

Alejandro Fuentes Barcelay

Tutores:

_____ _____

Ing. Sailyn Salas Hechavarria Ing. Luis Kayrumet Pérez Buigas

DATOS DE CONTACTO

Tutores:

Tutor principal: Ing. Sailyn Salas Hechavarria.

Especialidad: Ingeniero en Ciencias Informáticas.

Resumen: Profesora Asistente, ha dirigido diversas tesis de pre-grado. Es analista del centro VERTEX de la facultad 5. Es profesora principal de 3er año. Ha tenido diversas publicaciones tanto nacionales como internacionales.

Correo electrónico: ssalas@uci.cu

Tutor: Ing. Luis Kayrumet Pérez Buigas

Especialidad: Ingeniero en Ciencias Informáticas

Resumen del Tutor

Graduado en el 2014 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informática, trabaja como especialista en el centro VERTEX de la facultad 5.

Correo electrónico: kayrumet@uci.cu

Co-tutor: Manuel Alejandro Chang Fajardo

Especialidad: Ingeniero en Ciencias Informáticas

Resumen del Tutor

Graduado en el 2014 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informática, trabaja como especialista en el centro de la Facultad 6.

Correo electrónico: mchang@uci.cu

DEDICATORIA

Dedico este trabajo a mi familia por estar siempre ahí para mi y esperar siempre lo mejor de mí.

A mi papá y mi mamá por apoyarme en todo momento y darme la educación que hoy tengo.

A mi hermana Yoli que siempre me ha ayudado y por ser mi segunda mamá.

A mi amigo fallecido Ángel del Pino que de seguro estaría conmigo hoy disfrutando este momento, así como mis abuelos Petra y Antero que aunque no cuento físicamente con ellos son mi fuerza para seguir adelante cada día.

AGRADECIMIENTOS

A mis tutores por haber confiado en mi incluso cuando parecia imposible conseguir nuestro objetivo.

A mis padres por todo lo que han confiado en mi, y haber intentado hacer un hombre de bien.

A mis hermanos Andris, Tania, Bernabé y Yoli por siempre exigir de mi lo mejor y ser mi ejemplo a seguir.

A mis sobrinos Yania, Yacira, Yiliana y Geovani, por alegrarme la vida cada día que comparto con ellos.

A toda mi familia que de una forma u otra siempre me apoyaron e hicieron posible este sueño.

A mis amigos de la carrera sin ellos seria imposible llegar hasta aqui, Manolo, Julio, Hubert, Polo, Gladys, Yamila,Claudia, Tanya, Roger, Elizabeth, Henry, Frank, Yaikel, José Raúl, Ariel, Arian, y los búfalos con Joel al frente liderando como siempre.

A mi primo Yoanni por ser casi un hermano durante mi tiempo en la UCI ya que era lo más cercano a la zona con lo que contaba.

A todos mis hermanos del barrio Orlenis, Oriol, Juan Luis, Bamba, Tito, Karel, Manolito, Rainer, Julio el zapatero, entre otros socios de la zona, que me ayudaron e influyeron en mi vida en el barrio.

A Yaimi por ser mi hermanita blanca.

A Yarianna por ser mi grillito de la conciencia durante los últimos meses de la tesis.

RESUMEN

La Priorización de Requisitos de Software, implica la constitución de un proceso de decisión de naturaleza compleja. Este proceso es producto de la interacción de los involucrados quienes elaboran un juicio crítico y consensuado, con el fin de seleccionar en primera instancia el conjunto de requisitos que deberían ser implementados a lo largo del ciclo de vida del software. El ciclo de vida de un proyecto en la Universidad de las Ciencias Informáticas se realiza mediante el programa de mejora. Este permite priorizar requisitos pero no deja expresado en qué momento se podrán implementar los mismos. Por otra parte no se cuenta con una herramienta que utilice los métodos de priorización de requisitos clásicos.

Para dar solución a la situación planteada anteriormente se tomó como objetivo desarrollar una herramienta informática que permita la priorización de requisitos. Se escogieron las herramientas pertinentes que permitirán el desarrollo de la misma, estas fueron C++ como lenguaje de programación, *Qt Creator* como entorno de desarrollo y XML para guardar los reportes, así como metodología de desarrollo se eligió XP.

Al poner en marcha el uso de esta herramienta informática se permitirá la priorización de requisitos basado en un proceso de computación con palabras. Esta aplicación contribuye al desarrollo de software al colaboraren la planificación y selección de un conjunto ordenado requisitos de software.

Palabras claves: computación con palabras, involucrados, priorización, requisitos

TABLA DE CONTENIDOS

Introducción 9

Capítulo 1. Fundamentación Teórica 13

1.1 Priorización de Requisitos 13

1.2 Métodos de priorización de requisitos 15

1.3 Toma de decisiones 17

1.4 Computación con palabras 17

1.5 Modelos de computación con palabras 19

1.6 Operadores de Agregación 23

1.7 Metodologías y herramientas 25

1.8 Conclusiones Parciales 26

Capítulo 2 Análisis y diseño 27

2.1 Requisitos no funcionales 27

2.2 Fase de Exploración 28

2.3 Fase de planificación 31

2.4 Propuesta de solución 36

2.5 Patrón arquitectónico 37

2.6 Patrones de Diseño 38

2.7 Diseño de la solución propuesta 42

2.8 Conclusiones Parciales 46

Capítulo 3 Implementación y Prueba 48

3.1 Estándares de codificación 48

3.2 Implementación 51

3.3 Pruebas 55

3.4 Caso de estudio: Rehabilitador de Marcha 60

3.5 Conclusiones parciales 62

Conclusiones Generales 64
Recomendaciones 65
Referencias bibliográficas 65
Bibliografía 68
Anexos 73
Glosario de Términos 87

Introducción

Una de las industrias que más auge ha tomado en los últimos tiempos es la industria informática, se debe principalmente a la producción de software, en la que la priorización de requisitos es uno de los procesos de suma importancia en su elaboración (1). Debido al continuo incremento en el número de requisitos de software para los productos impulsados por el mercado hay una creciente necesidad de métodos capaces de dar prioridad a los requisitos candidatos. No todos los requisitos pueden ser atendidos simultáneamente y se debe tener en cuenta las limitaciones en cuanto a recursos que presente la empresa en cuestión (2). Por lo anteriormente planteado es importante para las organizaciones involucradas no solamente permitir a los clientes asignar prioridades a los requisitos y tomar decisiones acerca de ellos, sino también proporcionarles soluciones alternativas diferentes que permitan dar respuesta a sus necesidades.

Para atender las necesidades de los clientes es necesario realizar todo un trabajo que debe quedar plasmado en varios documentos. Uno de estos documentos es el de Especificación de Requisitos de Software mediante el cual se deja bien claro todo lo referente a los requisitos que serán atendidos durante la producción del software.

En el documento de Especificación de Requisitos de Software es importante realizar una efectiva priorización de requisitos, con el propósito de diferenciar cuáles son más significativos para el sistema y cuáles no, y el orden en que se deben implementar. El proceso de priorización de requisitos se basa en la toma de decisiones de un grupo de expertos teniendo en cuenta criterios que permiten realizar dicha acción. Los expertos tienen el peso fundamental en este proceso ya que de la decisión que estos determinen depende el éxito o no del proyecto.

La toma de decisiones puede ser vista como un proceso conformado por diferentes fases, tales como la recopilación de información, el análisis y la selección basada en los diferentes procesos mentales y el razonamiento que condujo a elegir una alternativa adecuada entre un grupo de alternativas posibles en una actividad determinada. En ocasiones, el proceso de resolución de un problema de toma de decisiones es sencillo utilizando un enfoque algorítmico, cuando ocurren estos casos entonces se está en presencia de los llamados problemas bien estructurados. Sin embargo, muchos problemas de decisión no pueden resolverse de esta manera porque las decisiones pueden estar relacionadas con los cambios del entorno.

Para solucionar este tipo de problemas es necesario modelar la incertidumbre a través del uso del enfoque lingüístico difuso (3), este se basa en la teoría de conjunto difuso (4) y en el concepto de variable lingüística(5). Estas variables son palabras o frases definidas en un lenguaje natural o artificial, que pueden representar la información lingüística. Utilizar información lingüística implica la necesidad del uso de computación con palabras (CW, por sus siglas en inglés), ya que las variables de entrada y salida se deben expresar en un dominio lingüístico (5).

En los últimos años en Cuba se han desarrollado iniciativas para informatizar la sociedad, dentro de las cuales se encuentra el surgimiento de la Universidad de las Ciencias Informáticas (UCI). La UCI organiza su producción de software a través de centros productivos. Uno de estos centros es el de Entornos Iterativos 3D (VERTEX), encargado de crear soluciones informáticas que permiten el desarrollo de herramientas con un alto nivel de complejidad tanto en tiempo como en recursos.

La **problemática de la investigación** radica en que:

En el centro VERTEX el desarrollo de los proyectos toman como guía el Programa de Mejoras, que propone un Expediente de Proyectos, donde se define el artefacto Especificación de requisitos, que tiene en cuenta la prioridad de requisitos por parte del cliente. En dicho artefacto:

- La priorización de los requisitos se realiza a criterio de los clientes sin una argumentación sólida.
- No se tiene en cuenta los criterios por los cuáles se deben regir los involucrados del proyecto para establecer en qué iteración van a ser implementados los requisitos.
- No se tienen en cuenta criterios que aportan en qué orden serán implementados los requisitos, por lo que se ve afectada la planificación del proyecto.
- Afecta el orden lógico de la implementación con sus correspondientes implicaciones negativas en el negocio.
- No existe de forma estructurada la utilización de métodos que apoyen el proceso de priorización de requisitos.

Estos factores impiden que la realización del software se desarrolle en el menor tiempo posible.

Se define como **problema a resolver**:

¿Cómo realizar la priorización de requisitos para la correcta planificación del cronograma en el desarrollo de software?

Tomando como **objeto de estudio** priorización de requisitos de software.

Campo de acción de priorización de requisitos de software basado en Computación con Palabras.

Objetivo general:

Desarrollar una herramienta informática para la priorización de requisitos de software basado en Computación con Palabras para lograr una correcta planificación del cronograma en el desarrollo de software.

Para dar cumplimiento al objetivo propuesto y dar solución a la situación problemática se proponen las siguientes tareas de investigación a cumplir:

- Caracterización del estado del arte de la priorización de requisitos para mejor entendimiento del objeto de estudio y el campo de acción.
- Definición de las tecnologías y lenguajes necesarios para el desarrollo de la aplicación.
- Definición de los requisitos del sistema.
- Diseño e implementación de las funcionalidades del sistema.
- Validación de la propuesta de solución para la evaluación de casos de estudios.

Métodos científicos e investigativos:

Para todo el proceso de investigación y elaboración de este trabajo se utilizaron los siguientes métodos científicos:

Teórico:

Histórico – Lógico: Método teórico mediante el cual se conocerá como ha sido la trayectoria histórica real, la evolución y desarrollo de los aspectos principales con los que cuenta la priorización de requisitos.

Analítico – Sintético: Este método teórico será utilizado en la investigación para buscar y analizar toda la documentación, permitiendo la extracción de las características más importantes de los métodos de computación con palabras que luego serán utilizadas para permitir la correcta priorización de requisitos.

Empíricos:

Consulta de fuentes de información: Método que permite tener acceso a la información disponible sobre el tema.

Observación: Método que permite conocer la realidad mediante la percepción directa de los objetos y fenómenos, es una manera de acceder a la información directa e inmediata sobre el proceso, fenómeno u objeto que está siendo investigado.

Estructura del trabajo:

El presente trabajo tiene la siguiente estructura: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos.

En el Capítulo 1: Fundamentación teórica. Mediante este capítulo se darán a conocer los fundamentos teóricos del trabajo de diploma. Además se mostraran los resultados del análisis y estudio del arte de las herramientas que permiten la priorización de requisitos tanto en el ámbito nacional como internacional.

En el Capítulo 2: Análisis y Diseño. En este capítulo se realizará la selección del lenguaje de programación, la metodología de software, el entorno de desarrollo integrado (IDE), los patrones arquitectónicos y a su vez la arquitectura del software, entre otras herramientas. Además de realizar los pasos a seguir para aplicar la metodología ágil Extreme Programming (XP).

En el Capítulo 3: Implementación y Prueba: Se establecerán los criterios para realizar la implementación y la validación del sistema desarrollado. Además se realizarán pruebas de aceptación al software para verificar el funcionamiento del mismo y de esa manera comprobar que todos los requisitos han sido correctamente implementados. Por otra parte se utiliza el método de caso de estudio para la validar la propuesta.

Capítulo 1. Fundamentación Teórica

En el presente capítulo se expondrán un conjunto de elementos que ayudan a definir los fundamentos teóricos de la investigación. Además se realizará un análisis del estado del arte, haciendo énfasis en los conceptos y algunos elementos importantes relacionados con la priorización de requisitos y el proceso de Computación con Palabras. También se realiza una revisión acerca de las herramientas existentes que permiten priorizar requisitos.

1.1 Priorización de Requisitos

La priorización de requisitos es una actividad crítica dentro de la Ingeniera de Requisitos (IR), ya que resulta de vital importancia realizar este proceso, porque mediante este se determinará el orden de los requisitos en la fase de desarrollo de software .Esta implica la constitución de un proceso de decisión de naturaleza compleja (6), ya que permite afrontar la toma de decisiones en los aspectos de la vida; en fin, se puede modelar como un problema de toma de decisiones (7) para arribar a una decisión constante. En cuanto a esta actividad han existido diversas concepciones de cómo entender dicha acción. Por ejemplo, Somerville (8) lo considera una actividad mediante la cual los requisitos de software más importantes son descubiertos. Mientras, Firesmith (9) lo define como un proceso que determina el orden de implementación de los requisitos del sistema o el equivalente al proceso que determina el orden de importancia de los requisitos en función de la intuición de los interesados.

El autor de la presente investigación define que la priorización de requisitos es la acción de ordenar los requisitos y de esta manera decidir cuál de estos se realizará primero, el que tiene mayor importancia para los involucrados del proyecto, entre otras decisiones que se pueden tener en cuenta una vez que se tenga bien definido este proceso.

1.1.1 Criterios y factores

Existen criterios o factores que han sido citados en diferentes trabajos y documentos de investigación con la intención de abordar el problema que implica realizar la acción de priorizar requisitos. Luego cuando se emprende el proceso de priorización, se identifican en los siguientes aspectos:

Importancia: Se refiere a un criterio de priorización, por intermedio del cual, los involucrados del proceso manifiestan para cada requisito en particular, su ordenación relativa respecto al presente aspecto. Sin

embargo, el criterio “importancia” utilizado por un involucrado en particular, puede identificarse por una serie de conceptos que no necesariamente resultan análogos a otro participante. En consecuencia, y tal cual lo manifiestan (6), el criterio debería ser previamente especificado para que los interesados puedan priorizar en forma homogénea, cada uno de los requisitos disponibles.

Penalidad: Se refiere a un criterio de priorización, por intermedio del cual, se establece la implicancia de no implementar un determinado requisito (6).

Costo: Se refiere a un criterio de priorización, por intermedio del cual, se contabilizan las erogaciones monetarias incurridas a los efectos de implementar una serie de requisitos candidatos. Por otra parte, constituye un aspecto que refleja la complejidad y calidad de los requisitos de fácil implementación (6).

Tiempo: Se refiere a un criterio de priorización, por intermedio del cual, los requisitos a implementar se valoran de acuerdo al tiempo necesario que se ha de incurrir, a los efectos de implementarlos en forma efectiva (6). Así mismo, y tal cual manifiesta (6), el tiempo incurrido depende de las características del proceso de desarrollo del software y de sus actividades concurrentes e independientes.

Riesgo / Volatilidad: Se refieren una serie de criterios de priorización, por intermedio de los cuales, los requisitos son valorables de acuerdo a su grado inherente de riesgo y probabilidad de cambio. Tales factores, inciden en el ciclo de desarrollo del producto, y sus manifestaciones más salientes implican retrasos en el cronograma, costos, presupuestos de recursos, entre otros. La información disponible de los requisitos a implementar, se constituye como el factor clave que podría mitigar los efectos del riesgo y la volatilidad (6).

Estrategia de Negocios: Se refiere a un criterio de priorización, por intermedio del cual, se valora la implementación del conjunto de requisitos, en función de la probabilidad de inserción del producto, de la existencia de competidores en el mercado, entre otros factores. En sí mismo, se establece como un conjunto de sub-criterios que son ponderables por la entidad encargada de desarrollar el producto (6).

Utilidad: Se refiere a un criterio de priorización, por intermedio del cual, se pondera el grado de satisfacción que la implementación de un determinado requisito, le reporta a un conjunto de interesados. Considerando las preferencias, se intenta ordenar el conjunto de requisitos disponibles de acuerdo a las expectativas reveladas por los participantes (6).

1.1.2 Definir criterios

Los criterios que se tienen en cuenta en esta herramienta, son los criterios definido en el modelo de priorización de requisitos basado en el proceso de computación con palabras desarrollado por la autora Sailyn Salas. A continuación se mencionan los criterios de priorización de requisitos:

- Complejidad técnica: grado en que se puede entender y verificar el diseño de un sistema o componente.
- Dependencia entre requisitos: relación existente entre requisitos.
- Importancia para los clientes: grado relativo de que tiene el requisito para los clientes.
- Costo: esfuerzo para implementar el requisito en el sistema de software.
- Nivel del riesgo: calificación en función de las consecuencias en el proceso del negocio por su implementación.

1.2 Métodos de priorización de requisitos

Asignación Numérica

Esta técnica asigna un valor (escala) a cada requisito, adicionalmente los requisitos se dividen en tres grupos: Obligatorios, deseables y no esenciales. Los participantes asignan a los requisitos un número dentro de la escala de 1 a 5 indicando la importancia de cada uno de ellos. La clasificación final es el promedio de las clasificaciones de todos los participantes para cada uno de los requisitos. Los números en la escala tienen el siguiente significado (10):

- El cliente no lo necesita.
- No es importante.
- Bastante importante (Si lo tiene el cliente lo agradece).
- Es muy importante (El cliente lo desea).
- Es obligatorio (El cliente no puede prescindir de él).

Ordinal (Ranking)

Como el método Asignación Numérica, el presente procura una ordenación de los requisitos pero sin agruparlos en categorías. Es decir, se ordenan de 1 a N considerando el nivel “1” como el requisito más importante, y el “N” como el de menor importancia (11) (12). Dada su naturaleza, el presente método

procura que cada requisito pertenezca a una categoría de prioridad particular. En consecuencia, no permitirá establecer la diferencia relativa que existiría entre dos o más requisitos, con lo cual, la priorización relativa resulta inexistente (13).

Método de los 100 dólares

El presente método, también denominado Voto Acumulado ("Cumulative Voting") consiste en asignar una suma hipotética de dinero a cada involucrado, para que cada uno de ellos procure una asignación prioritaria del "dinero" hacia el conjunto de requisitos disponibles (14) (13) (11). En función de las asignaciones individuales, se tabulan las distribuciones en un diagrama de frecuencias relativas, indicando en consecuencia, el nivel de prioridad de cada uno de los requisitos (en porcentaje). Uno de los problemas que se le han imputado a este método, se refiere a la extensión del conjunto de requisitos que han de ser analizados, con lo cual, Regnelly otros ha comprobado en (15) la eficacia de incrementar "la cantidad de dinero distribuible" a los efectos de poder valorar conjuntos de requisitos más extensos.

Proceso de Jerarquía Analítica

El método AHP fue desarrollado por Saaty (6) (16) (17) (18) y se ha considerado como uno de los enfoques más utilizado cuando resulta necesario tomar una decisión, respecto a la selección y priorización de un determinado conjunto de alternativas, teniendo en consideración, el juicio provisto por un conjunto de sujetos (19) (20) (21).

El método finalmente fue aplicado por primera vez en el área de priorización de requisitos por Karlsson en (22). Si bien fue ampliamente criticada la metodología, ya que la existencia de requisitos dependientes y la inconsistencia en el juicio valorativo de los involucrados, relativizan la robustez de los resultados ofrecidos por el método, se ha convertido en un enfoque que ha evolucionado como un componente metodológico clave dentro de la Priorización de Requisitos de Software.

Los métodos analizados no se tienen en cuenta en la propuesta, pues presentan inconveniente. Los dos métodos iniciales (Asignación Numérica y Ordinal) involucran aspectos subjetivos y poco precisos que conllevan a que el proceso se torne engorroso. Sin embargo los dos últimos presentan el siguiente problema: cuando se tiene una lista extensa de requisitos la utilización de los mismos no es factible. Es por esta razón que el trabajo de diploma se enmarca en la priorización de requisitos a través de un proceso computación con palabras.

1.3 Toma de decisiones

La toma de decisión es un proceso habitual para los seres humanos en muchas actividades del mundo real como la ingeniería, organización, finanzas, medicina, entre otros campos de la vida diaria (23). Debido a esta variedad de disciplinas, los problemas de decisión se han clasificado en la teoría de decisión atendiendo a su estructura y elementos. De esta manera se tiene que la toma de decisiones es un proceso intencional que combina el análisis de información, la confrontación de alternativas, la valoración de las opciones y finalmente se toma la decisión. El proceso de tomar decisiones es la esencia de la planeación, entonces se puede decir que la toma de decisiones se puede concebir como (24):

- Establecimiento de premisas
- Identificación de alternativas
- Evaluación de alternativas en términos de la meta propuesta
- Elección de una alternativa

El proceso de tomar decisiones se forma con el uso de métodos estructurados y coherentes con el campo en el que se está trabajando. Un método se funda en datos, en recabar información de calidad, verificarla y contrastarla con otras del campo de producción específico y de otros. Para tomar una decisión, no importa su naturaleza es necesario conocer, comprender y analizar un problema, para así poder darle solución. En algunos casos por ser tan simples y cotidianos, este proceso se realiza de forma implícita y se soluciona muy rápidamente, pero existen otros casos en los cuales las consecuencias de una mala o buena elección pueden tener repercusiones en la vida.

Este proceso supone un análisis que requiere de un objetivo y una comprensión clara de las alternativas mediante las que se puede alcanzar dicho objetivo. Además de comprender la situación que se presenta se debe analizar, evaluar, reunir alternativas y considerar las variables, comparar varios cursos de acción y finalmente seleccionar la acción que se va a realizar.

1.4 Computación con palabras

La computación con palabras se basa en la lógica difusa y fue introducida por Lotfi Asker Zadeh en 1965. La metodología de la computación con palabras se basa en identificar palabras que representen cantidades y luego se les relaciona con conjuntos difusos. Bajo el contexto de CW a estos conjuntos se les conoce como variables lingüísticas.

Muchos autores exponen que es la CW, a continuación se citan algunos ejemplos:

Cuando un problema se resuelve utilizando la información lingüística, implica la necesidad de la Computación con Palabras (CW) (25).

Para Mendel y otros actores en (26) la definen como la "metodología para el razonamiento y la toma de decisiones haciendo uso de información descrita en lenguaje natural".

La Computación con palabras o *Computing with words* (CW) es una metodología auxiliar al procesamiento del lenguaje natural. Ésta utiliza números que representan cantidades que luego serán expresadas por medio de palabras, para realizar operaciones aritméticas. El resultado de estas operaciones son nociones o aproximaciones; sin embargo, esta metodología permite obtener resultados en algunas situaciones donde los cálculos precisos son imposibles.

El autor define a la CW como una metodología que permite mediante el uso de información descrita en lenguaje natural obtener resultados donde los cálculos precisos son imposibles.

1.4.1 Pasos y limitaciones de la computación con palabras

La idea básica que subyace al concepto de CW es que, en general, la información se transfiere restringiendo los valores que pueden tomar las variables lingüísticas (27).

Pasos a tener en cuenta para que se cumpla el concepto de CW.

1. La información dada se representa como una colección de proposiciones expresadas en lenguaje natural.
2. Cada proposición se ve como una restricción implícita sobre una variable implícita: la proposición expresada en lenguaje natural limita los significados posibles de la variable.
3. Por necesidades del cálculo, es necesario hacer explícitas esas restricciones.
4. Una vez que las restricciones implícitas en las premisas de partida se hacen explícitas mediante el uso de formas canónicas, se pueden usar las reglas de inferencia de la lógica para propagar las restricciones de las premisas a las conclusiones.
5. Finalmente, las restricciones inducidas se traducen a un lenguaje natural.

Algunos de los problemas más representativos que se tratan en este campo son los siguientes(28):

1. Solución de un sistema de ecuaciones lineales con coeficientes lingüísticos en lugar de numéricos.
2. Operaciones sobre funciones definidas mediante reglas difusas con valores no numéricos.
3. Cálculo de probabilidades lingüística.
4. Razonamientos con silogismos.

1.5 Modelos de computación con palabras

El uso del enfoque lingüístico difuso y de la información lingüística implica la realización de procesos de computación con palabras. Existen dos modelos computacionales inicialmente utilizados en el enfoque lingüístico difuso para realizar dichas operaciones: el modelo basado en el principio de extensión y el modelo simbólico. Los cuales son también conocidos como modelos computacionales clásicos (29).

1.5.1 Modelo Basado en el Principio de Extensión

Este modelo computacional, se caracteriza por utilizar funciones de pertenencia asociadas a la semántica de los términos lingüísticos y de esta forma poder realizar las operaciones con las palabras utilizando la aritmética difusa expresada en el principio de extensión. El uso de la aritmética difusa produce un incremento de la vaguedad en los resultados de computación con palabras, ya que los números difusos obtenidos de operar con etiquetas lingüísticas no suelen coincidir con los números difusos que representan la semántica de los términos lingüísticos iníciales. En este principio, tanto las relaciones de igualdad e inclusión como las operaciones de complementación, intersección y unión definidas para conjuntos difuso se reducen a las definiciones clásicas en conjuntos ordinarios (30).

1.5.2 Modelo Simbólico

El modelo Simbólico es otro de los modelos para operar con palabras. El cual utiliza la estructura ordenada del conjunto de términos lingüísticos $S = \{s_0, \dots, s_g\}$ donde $s_i < s_j$ si i < j, para llevar a cabo los procesos de computación. En este modelo computacional los resultados intermedios de las operaciones son valores numéricos $Y \in [0, g]$, que no tienen ninguna interpretación semántica ni sintáctica, por lo que deben ser aproximados en cada paso del proceso computacional mediante una función de aproximación app 2: [0, g] → {0, . . . , g}, que obtiene un valor numérico, el cual indica el índice del término lingüístico asociado a dicho resultado en el conjunto de términos lingüístico inicial s que es conjunto de términos lingüísticos app 2: (γ) ∈ S (31).

1.5.3 Modelo lingüístico de 2-tuplas

Este modelo fue presentado por Herrera y Martínez en (25) con el objetivo de lograr una mejor precisión de los modelos de computación con palabras. Por otra parte se encarga de expresar de forma simbólica cualquier resultado del universo del discurso. También se ha utilizado obteniendo resultados satisfactorios en el tratamiento de la información lingüística multigranular (32),lingüística no balanceada e información heterogénea (numérica, intervalar, lingüística) . En el modelo 2-tupla, el modelado de la información lingüística está basado en el concepto de traslación simbólica (33). Para mejor entendimiento se mostrará su modelo de representación de la información lingüística, para luego presentar su modelo computacional que cuenta con los operadores de comparación y agregación.

- **Modelo de representación**

Este modelo se basa en el concepto de traslación simbólica.

Definición 1: La traslación simbólica de un término lingüístico $s_i \in S = \{s_0, \dots, s_g\}$ es un valor numérico definido en $[-0.5, 0.5]$ lo que representa la "diferencia de información" entre una cantidad de información $\beta \in [0, g]$obtenida de una operación simbólica y el índice del término lingüístico más cercano.

La definición expresada anteriormente fue el punto de partida para que Herrera y Martínez llegaran a un nuevo concepto totalmente novedoso. Este usa como base la representación de una 2-tupla, (s_i, α), donde $s_i \in S = \{s_0, \dots, s_g\}$ representa la etiqueta lingüística y α es el valor que representa la traslación simbólica. Además con este modelo se podrá realizar un conjunto de transformaciones entre valores numéricos y2-tuplas para facilitar el proceso de computación con palabras.

Definición 2: Sea $S = \{s_0, \dots, s_g\}$un conjunto de términos lingüísticos. La 2-tupla asociada a S es definida como$\langle S \rangle = S \times [-0.5,0.5]$.

La función $\Delta: [0, g] \rightarrow \langle S \rangle$es mediante$\Delta(\beta) = (s_i, \alpha), con \begin{cases} i = round\ (\beta), \\ \alpha = \beta - i \end{cases}$, donde round es el operador de redondeo, $s_i$es la etiquetacon índice cercano aβ y α es el valor de traslación simbólica.

Ejemplo 2:

El funcionamiento de la función Δ que se acaba de definir puede observarse a continuación. Supóngase una operación de agregación simbólica sobre etiquetas valoradas en el conjunto de términos lingüísticos S = {Nada(N); Muy Bajo (MB); Bajo (B); Medio (M); Alto(A); Muy Alto (MA); Perfecto (P)} que obtiene como

resultado de dicha operación un valor$\beta = 2{:}8$, entonces la representación de esta información utilizando el modelo 2-tupla lingüístico que se representa en la Figura 1.

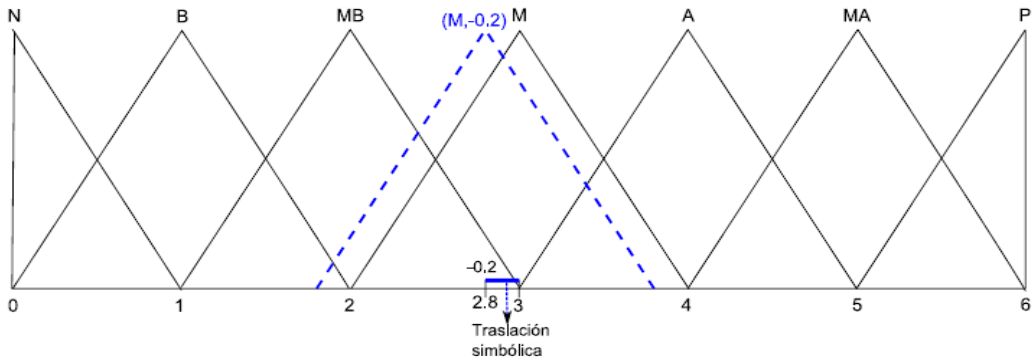


Figura 1: Ejemplo de una operación de Traslación Simbólica.

No se puede obviar el hecho de que la funciónΔ es biyectiva y $\Delta^1{:}\langle S\rangle \to [0,g]$es definida mediante$\Delta^1(S_i,\alpha) = i+\alpha$.De esta manera se forma la 2-tupla de $\langle S\rangle$la cual es identificada con el valor numérico en el intervalo$[0,g]$.

- **Modelo computacional**

Conjuntamente al modelo de representación de información lingüística, Herrera y Martínez definieron el modelo computacional basado en las funciones de trasformación Δ,Δ^1. Según (34) se definieron los operadores de comparación, negación, y varios de agregación que se le harán alusión a continuación.

- **Operador de comparación**

El operador de comparación de información lingüística que se representa mediante el modelo de 2-tupla se realiza utilizando un orden lexicográfico (33). Sea$(s_k,a_1)\ y\ (s_l,a_2)$ dos 2-tuplas cada una representando la cantidad de información, entonces:

Sean$(s_k,a_1)\ y\ (s_l,a_2)$ dos 2-tuplas, cada una representando una cantidad de información, entonces:

- Si k < l entonces($s_k,\ \alpha_1$) menor igual que ($s_l,\ \alpha_2$)
- Si k = l entonces
  a) Si $\alpha_1 = \alpha_2$, entonces ($s_k,\ \alpha_1$) y ($s_l,\ \alpha_2$) representan la misma información.
  b) Si $\alpha_1 < \alpha_2$, entonces($s_k,\ \alpha_1$)es menor que($s_l,\ \alpha_2$).

c) Si $\alpha_1 > \alpha_2$, entonces(s_k, α_1)es mayor que(s_l, α_2).

- **Operador de negación**

El operador de negación sobre una 2-tupla se define como:$Neg\ (s_i, \alpha) = \Delta\left(g - \left(\Delta^{-1}(s_i, \alpha)\right)\right)$,siendo g + 1 la cardinalidad del conjunto de etiquetas S, donde $S = \{s_0, \ldots, s_g\}$.

- **Operador de agregación**

Este operador consiste en obtener un valor colectivo que exprese la información de un conjunto de valores marginales. En la literatura, se puede encontrar numerosos operadores de agregación que permiten combinar la información de acuerdo a distintos criterios. Cualquiera de estos operadores puede ser fácilmente extendido para trabajar con 2-tuplas usando funciones $\Delta\ y\ \Delta^{-1}$, que transforman valores numéricos en 2-tupla y viceversa sin pérdida de información (33). El resultado de realizar esta operación debe cumplir con la característica de ser consistente con la representación de los valores de entrada entonces el resultado de la agregación de 2-tuplas debe ser una 2-tuplas. Además este operador cuenta a su vez con un conjunto de operadores que fueron definidos en (34).

3.1) Media aritmética

Definición 9: Sea x =$x_1, \ldots, x_n$un conjunto de valores numéricos para una variable x. La media aritmética$\bar{x}$, se obtiene dividiendo la suma de todos los valores por su cardinalidad:

$$\bar{x}(x_1, \ldots, x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i \quad [1]$$

La media aritmética simboliza el concepto intuitivo de centro del conjunto de valores o punto de equilibrio. Se puede extender este operador para obtener un operador equivalente para trabajar la información lingüística representada en el modelo de 2-tuplas. Lo anterior expresado se puede lograr mediante la siguiente definición:

Definición 3: Sea $x = \{(r_1, \alpha_1), \ldots, (r_m, \alpha_m)\}$ un conjunto de 2-tuplas, su media aritmética se calculará con el operador media aritmética extendida$\bar{x}^e$, que es definido como:

$\bar{x}^e[(s_i, \alpha_i), \ldots, (s_n, \alpha_n)] = \Delta\left(\sum_{i=1}^{n} \frac{1}{n} \Delta^{-1}(s_i, \alpha_i)\right) = \Delta\left(\frac{1}{n} \sum_{i=1}^{n} \beta_i\right)$ [2]

1.6 Operadores de Agregación

Los aperadores de agregación juegan un papel importante, ya que son un tipo de función matemática empleada para la fusión de la información. Combinan n valores en un dominio D y devuelven un valor en ese mismo dominio (35). Denominando esas funciones $\mathbb{C}$ (35), los operadores de agregación son funciones de forma:

$$\mathbb{C}: N^n \rightarrow N \text{ [3]}$$

Algunos de los operadores de agregación más empleados son los siguientes:

- **Media ponderada**: la media ponderada (WA, por sus siglas en inglés) es uno de los operadores de agregación más empleados en la literatura. La media ponderada se define de la siguiente forma:
  Definición 1: Un operador WA tiene asociado un vector de pesos V, con $v_i \in [0,1]$ y $\sum_1^n v_i = 1$, teniendo la siguiente forma:
  $$WA(a_1, .., a_n) = \sum_{i=1}^{n} v_i a_i \text{ [4]}$$
  , donde $\mathrm{v_i}$ representa la importancia/relevancia de la fuente de datos a_i.

- **OWA** (Ordered Weighted Aggregation o traducido al español media ponderada ordenada): Operador creado por Yager, además es un operador ponderado, en el cuál, los pesos no están asociados a un valor predeterminado sino a una posición predeterminada.
  Definición 11: Sea $A = \{a_1, .., a_n\}$un vector de valores numéricos y $W = (w_1, .., w_n)$ un vector de peso asociado, tal que, (i) $w_i \in [0, 1]$ y (ii) $\sum w_i = 1$.El operador OWA, F, se obtiene como:
  $\mathrm{F(a_1, \ldots, a_n)} = \sum_{\mathrm{j=1}}^{\mathrm{n}} \mathrm{w_j b_j}$, donde b_j es el j-esimo mayor valor del conjunto A.
  Para trabajar con 2-tuplas el operador OWA fue extendido como se presentará a continuación: Definición 8 Sea$A = \{(r_1, \alpha_1), \ldots, (r_m, \alpha_m)\}$ un conjunto de 2-tuplas y $W = (w_1, .., w_n)$un vector de pesos asociado que satisface que (i) $w_i \in [0, 1]$y (ii) $\sum w_i = 1$. El operador OWA extendido F^e para combinar 2-tuplas actúa como:
  $F^e\big((r_1, \alpha_1), \ldots, (r_m, \alpha_m)\big) = \Delta\left(\sum_{j=1}^{m} w_i \beta_j^*\right)$ [5]

- D-OWA El operador OWA dependiente (D-OWA)(36) penaliza con una menor fiabilidad los elementos que se alejen de la media aritmética (μ), disminuyendo la influencia de estos valores en la agregación según la siguiente expresión matemática:

$$\mathrm{D-OWA}(a_1, \ldots, a_n) = \frac{\sum_{j=1}^{n} s(a_j, \mu) a_j}{\sum_{j=1}^{n} s(a_j, \mu)} \quad [6]$$

donde $s(a_j, \mu)$ es una función de similitud entre el elemento a_j y μ (37).

- El operador D-OWAWA permite combinar las ventajas del operador D-OWA y WA en el cálculo de medidas compuestas.

Definición: Un operador D-OWAWA es una función $\mathbb{R}^n \to \mathbb{R}$ de dimensión n si tiene un vector de ponderaciones$W$asociado, con$W = \sum_{j=1}^{n} w_j = 1$ y $w_j \in [0,1]$tal que:

$$D - OWAWA(a_1, a_2, a_3 \ldots a_n) = \sum_{i=0}^{n} \hat{v}_i a_i \quad [7]$$

Donde$a_i$tiene asociado una ponderación$\hat{v}_i$

$$\hat{v}_i = \beta w_i + (1-\beta)\omega_i \text{ ,con } \beta \in [0,1] \quad [8]$$

ω_i , corresponde al i-ésimo elemento del vector de pesos calculados a partir del operador D-OWA.

En el trabajo de diploma se utiliza el operador de agregación 2TD-OWAWA que propone la tutora Sailyn Salas, el cual unifica los operadores D-OWA y WA para el método 2-tuplas.

Definición: Siendo$X = \{(s_i, a_i), \ldots, (s_n, a_n)\}$un conjunto de varias 2-tuplas lingüísticas, y T un vectorde peso de dimensión n relacionado con el operador D-OWA de forma tal que $\sum_{j=1}^{n} T_j = 1 \; y \; T_j \in [0,1]$, y K un vector de peso de dimensión n relacionado con el operador WA, con$\sum_{j=1}^{n} K_j = 1 \; y \; K_j \in [0,1]$tal que:

$$\text{2TD-OWAWA}((s_1, a_1), \ldots, (s_m, a_m)) = \Delta^{-1}\left(L \sum_{j=1}^{m} T_j \beta_j + (1-L) \sum_{j=1}^{m} K_j \beta_j\right) \quad [9]$$

β_j, es el j-ésimo mayor elemento dea_i, $L \in [0,1]$ indica en qué medida se valoran los operadores en la agregación final.

Se seleccionó el modelo lingüístico basado en 2-tuplas por las razones que a continuación se explican:

Los modelos clásicos mencionados anteriormente presentan las siguientes limitaciones:

- Para obtener un resultado expresado en el dominio lingüístico de expresión inicial, hay que realizar procesos de aproximación que implican pérdida de información y precisión en los resultados (29).
- El modelo Simbólico puede obtener resultados precisos, pero expresados en un dominio de expresión distinto al conjunto de términos iníciales y difíciles de entender (29).

1.7 Metodologías y herramientas

1.7.1 Metodología utilizada

Para el desarrollo de la herramienta se va a utilizar la Programación Extrema (XP, por sus siglas en inglés), una metodología de procesos ágiles para el desarrollo de software. Es de las metodologías más exitosas debido a que se centra en potenciar las relaciones entre los desarrolladores para lograr el éxito, promoviendo el trabajo en equipo y la comunicación con el cliente, lo cual propicia un buen entorno de trabajo (38). Algunas de las características que los distinguen son:

- Desarrollo iterativo e incremental.
- Integración del equipo de programación con el usuario.
- Propiedad del código compartida.
- Corrección de errores.

1.7.2 Lenguaje de programación

Para realizar la programación de la herramienta se decidió escoger como lenguaje de programación C++ ya que este es un lenguaje orientado a objeto. Además entre sus características principales tiene que es rápido y eficiente. Los programas tendrán menos errores porque C++ usa una sintaxis y chequeo de tipos más estricto. También cuenta con una gran ventaja que es de propósito general puesto que con él se pueden realizar aplicaciones desde sistemas operativos y compiladores hasta aplicaciones de bases de datos, procesadores de texto, juegos, entre otras (39).

1.7.3 Entorno de desarrollo integrado

Qt Creator fue el Entorno de desarrollo integrado (IDE) escogido para dar cumplimiento a los objetivos propuestos para dar feliz término al producto. Dicha elección fue abalada ya que este IDE es multiplataforma o sea un proyecto puede ser desarrollado en Microsoft Windows, Mac OS X o en Linux. Además este cuenta con un editor de código en C++ que fue el lenguaje de programación seleccionado para el desarrollo de la

herramienta a implementar, permite el trabajo con bases de datos, cuenta con una interfaz gráfica de usuario y diseñador de formularios (40).

1.7.4 Lenguaje XML

Otro leguaje que se utilizara es XML, siglas en inglés de *eXtensible Markup Language* ('lenguaje de marcas extensible'), es un lenguaje utilizado para almacenar datos de forma legible. Permite definir la gramática de lenguajes específicos (de la misma forma que HTML) para estructurar documentos grandes(41).

1.8 Conclusiones Parciales

Del presente capítulo se concluye que:

- Se seleccionó el método 2-tuplas, ya que los métodos de priorización estudiados presentan inconveniente a la hora de llevar a cabo este proceso.
- Se seleccionó el método 2TD-OWAWA para unificar los juicios de los expertos y así lograr una mayor fiabilidad e interpretabilidad a la hora de priorizar requisitos.
- Se decidió utilizar las herramientas antes mencionada, ya que estas son por las que se rige el centro VERTEX.

Capítulo 2 Análisis y diseño

En el presente capítulo se evidencian los elementos relacionados con la solución desarrollada, teniendo en cuenta varios aspectos como la metodología, el lenguaje de programación y el entorno de desarrollo integrado que se utilizó. Además se hace mención de las características generales que tiene la herramienta que se desarrolló, también se definen los requisitos del sistema, tanto los funcionales como los no funcionales. Otro aspecto que se aborda es la arquitectura que se seleccionó para la implementación de la herramienta, los patrones de diseño de programación utilizados entre otros aspectos importantes.

2.1 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Estos no tienen ninguna relación con las funciones específicas del sistema porque se encargan de propiedades que el software debe cumplir como fiabilidad, tiempo de respuesta, capacidad de almacenamiento, entre otras (42).

2.1.1 Interoperabilidad

RNF1. El sistema debe cumplir con ser multiplataforma, por lo que el mismo tiene que funcionar tanto en Linux como en Windows. Esto minimiza las molestias por parte del cliente lo que colabora de forma directa en el compromiso de estos con el software.

2.1.2 Hardware

RNF2.Para el correcto funcionamiento del sistema se necesita un procesador Dual Core 2.2. Además una Memoria RAM de 512 MB.

2.1.3 Rendimiento

RNF3. El tiempo de respuesta será de 0,5 segundo.

2.1.4 Usabilidad

RNF4.Fácil acceso a las funcionalidades para la priorización de requisitos.

RNF5. La herramienta es sencilla de usar, ya que comprende una fácil navegación por los diferentes contenidos que en esta intervienen, teniendo como premisa fundamental la adecuada estructuración y

organización de la información. Las características anteriormente descritas se tienen en cuenta para que el usuario tenga el conocimiento y el dominio requerido del lugar del sistema en el que se encuentra y no se pierda.

2.1.5 Restricciones en el Diseño e implementación

RNF6.Lenguaje de programación C++ bajo el IDE Qt Creator 5.3.0.

RNF7. La biblioteca de desarrollo utilizada es Qt en su versión 5.3.0.

2.2 Fase de Exploración

Esta fase es la primera que se debe realizar cuando se usa la metodología XP. En la misma los clientes escriben las historias de usuario, de lo que ellos quisieran incluir para la primera entrega del software. En cada plantilla se realiza una descripción de las características que deben ser incluidas en la herramienta. Al mismo tiempo el equipo del proyecto se familiariza con las herramientas, la tecnología y las prácticas que utilizarán durante la realización del producto. La fase de exploración toma entre unas cuantas semanas a unos cuantos meses, dependiendo de que tanto los programadores conocen la tecnología (43).

2.2.1 Historias de usuario

En las historias de usuarios se presenta una breve descripción del comportamiento del sistema, se debe emplear una terminología asequible para cliente sin lenguaje técnico. Estas se deben realizar una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. Estas deben proporcionar sólo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere la implementación de la historia, difiere de los casos de uso porque son escritos por el cliente, no por los programadores, empleando terminología del cliente (44). En el presente trabajo se expondrán las principales historias de usuarios las restantes se encuentran en los anexos A.

Las Historias de Usuario tienen tres aspectos:

Tarjeta: en ella se almacena suficiente información para identificar y detallar la historia.

Conversación: cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación)

Pruebas de Aceptación: permite confirmar que la historia ha sido implementada correctamente.

2.2.2 Relación de las historias de usuario

Tabla 1. Historia de Usuario 1

Historia de Usuario	
Número : 1	Nombre de HU : Insertar requisito
Fecha : 23-02-2015	Usuario : Cliente
Prioridad en el Negocio : Alta	Riesgo de Desarrollo : Alto
Puntos Estimados : 2/3	Iteración Asignada :1
Descripción: El sistema debe permitir al usuario insertar los requisitos que desee priorizar. Para realizar esta acción debe introducir al sistema el nombre del requisito, número que lo identifica y la descripción de este.	
Observaciones:	

Tabla 4. Historia de usuario 4

Historia de Usuario	
Número: 4	Nombre de HU: Seleccionar criterio
Fecha: 25-03-2015	Usuario: Cliente
Prioridad en el Negocio: Alta	Riesgo de Desarrollo: Alto

Puntos Estimados: ¾	Iteración Asignada: 2
Descripción: El sistema debe permitir al usuario seleccionar los criterios por lo que se va a guiar para realizar la priorización. Para seleccionar se debe tener en cuenta el valor de la importancia que se debe dar de 1 a 5. Observaciones:	

Tabla 6. Historia de usuario 6

Historia de Usuario	
Número: 6	Nombre de HU: Insertar experto
Fecha : 02-03-2015	Usuario: Cliente
Prioridad en el Negocio: Alta	Riesgo de Desarrollo: Alto
Puntos Estimados: 2/3	Iteración Asignada: 1
Descripción: El sistema debe permitir al usuario insertar los expertos que desee. A la hora de insertar se deben llenar todos los datos correspondientes a los expertos: nombre, apellidos, número de experto, teléfono, centro, correo electrónico, experiencia que se debe dar entre 1 y 5. Observaciones:	

Tabla 10. Historia de usuario 10

Historia de Usuario	
Número: 10	Nombre de HU: Mostrar orden por requisito
Fecha: 31-03-2015 Prioridad en el Negocio: Alta Puntos Estimados: 1/2	Usuario: Cliente Riesgo de Desarrollo: Alto Iteración Asignada: 2
Descripción: El sistema debe permitir al usuario la visualización de cómo quedan repartidos los requisitos por orden de importancia. Para esto luego de realizar la priorización se debe dar un clic sobre la columna Prioridad que se encuentra en la pantalla principal. Entonces después de realizar las acciones anteriores se podrá visualizar el orden en el cual los requisitos deben ser implementados.	
Observaciones:	

2.3 Fase de planificación

En esta fase se establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto (45).

Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Además el equipo de desarrollo debe mantener un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de

usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación (45).

2.3.1 Estimación de esfuerzo

La estimación de esfuerzo se puede confundir con la estimación de tiempos. El esfuerzo se refiere a la suma de los tiempos que le dedicarán los diferentes recursos a cierta actividad o al proyecto. Se mide en horas/hombre, días/hombre, semanas/hombre y otros. No importa que el trabajo se haga de forma secuencial por un solo recurso o en paralelo por diferentes personas. Se suman los tiempos de cada uno de ellos para obtener el esfuerzo total (46). A continuación se presenta el resultado de la estimación de esfuerzo que se realizó a cada historia de usuario.

Tabla 15. Estimación de esfuerzo

Historias de Usuario	Puntos de Estimación
Insertar requisito	2/3semanas
Modificar requisito	2/3semanas
Eliminar Requisito Seleccionar Criterio Seleccionar importancia de los criterios	2/3semanas ¾ semana 2/3 semana
Insertar Experto Modificar Experto	2/3semanas 2/3semanas

Eliminar Experto	2/3semanas
Mostrar priorización	2 semanas
Mostrar orden por requisito	½ semanas
Guardar XML	1 semanas
Importar XML Seleccionar clasificación	1 semanas ½ semanas
Visualizar matriz	½ semanas

2.3.2 Plan de iteraciones

En esta fase se incluyen varias iteraciones donde el plan de entrega de cada iteración no excederá las 5 semanas, después de identificadas y descritas las HU y estimado el esfuerzo para la realización de cada una de ellas, se procede a la planificación de la etapa de implementación de la herramienta. Se especifica cuáles son las HU que se implementarán en cada iteración y se determinan las posibles fechas de entrega. Después de haber definido y estimado las HU se tomó la decisión de realizar 3 iteraciones, las cuales se detallan a continuación.

Iteración # 1

La primera iteración tiene como objetivo implementar las HU #1, #2, #3, #6, #7 y #8. Con prioridad para el cliente de Alta para todas las historias de usuario atendidas en la presente iteración. Las HU insertar, modificar y eliminar requisitos son la base para lograr la implementación de las siguientes iteraciones. Esta iteración se realizará en 4 semanas para implementar todas las tareas. Como resultado de esta iteración se obtiene una primera versión de la herramienta la cual será presentada al cliente para la retroalimentación con el equipo de desarrollo para dar paso a la siguiente iteración.

Iteración #2

Para cumplir con los objetivos de esta se debe implementar las HU #4, #5, #9, #10, #13 y #14. Esta fase cuenta con 5 semanas para su realización. Tiene prioridad Media para el cliente la HU #5 y Alta para todas las demás historias de usuarios implementadas en esta iteración. Al finalizar se obtendrá una segunda versión del sistema propuesto y se le hará llegar al cliente la iteración anterior junto con la presente para la aprobación o cambios pertinentes con el cliente.

Iteración # 3

La iteración # 3 y última corresponde a la HU #11, #12 la cual tiene como objetivo de asegurar el almacenamiento de los datos que son introducidos al sistema. En esta iteración se completará el 100% de las funcionalidades del sistema propuesto para ser probada por el cliente en su totalidad.

Las iteraciones a desarrollar presentan un nivel medio y alto de complejidad para el desarrollo con alta prioridad para el cliente. Se dispondrán de 12 semanas para desarrollar todas las iteraciones propuestas.

2.3.3 Plan de duración de las iteraciones

Para lograr una mayor organización del trabajo se crea un plan de duración de las iteraciones; el mismo tiene como objetivo mostrar la duración de cada iteración así como el orden en que serán implementadas las historias de usuarios en cada una de ellas.

Tabla 16. Plan de duración de las iteraciones

No Iteración	Historias de Usuario	Duración total de iteraciones
Iteración # 1	Insertar Requisito	4 semanas
	Modificar Requisito	
	Eliminar Requisito	
	Insertar Experto	
	Modificar Experto	
	Eliminar Experto	

Iteración # 2	Seleccionar Criterio	5 semanas
	Mostrar importancia de los criterios	
	Mostrar priorización	
	Mostrar el orden por requisitos	
	Seleccionar complejidad	
	Visualizar la matriz	
Iteración # 3	Importar XML	2 semanas
	Guardar XML	

2.3.4 Plan de Entrega

Para lograr una mayor organización del trabajo se crea un plan de duración de las iteraciones; el mismo tiene como objetivo mostrar la duración de cada iteración así como el orden en que serán implementadas las historias de usuarios en cada una de ellas.

Tabla 17. Plan de entrega de iteraciones

No Iteración	Duración	Fecha Inicio	Fecha Final
Iteración # 1	4 semanas	20 – 02 – 2015	18 – 03 – 2015
Iteración # 2	5 semanas	24 – 03 – 2015	28 – 04 – 2015
Iteración # 3	2 semanas	30– 04 – 2015	14 – 05 – 2015

2.4 Propuesta de solución

Se propone desarrollar una herramienta capaz de priorizar requisitos, donde se expresará además el orden en el cual se debe desarrollar dichos requisitos. El sistema debe gestionar tanto los requisitos como los expertos permitiendo realizar las acciones de insertar, modificar y eliminar. Los criterios serán tratados de forma estática, ya que los mismos no estarán sujetos a cambios al ser criterios seleccionados por un grupo de expertos en el tema. Vale destacar que una vez seleccionado los criterios a tener en cuenta en la priorización, se debe definir un valor de importancia entre 1 y 5 para cada criterio seleccionado.

El software cuenta con un área de trabajo donde se encontrará una matriz que relaciona los siguientes elementos: criterios y requisitos. En la misma están definidas las complejidades (Muy Alto, Alto, Medio, Bajo, Muy Bajo), mediante las cuales los expertos expondrán sus juicios. En cuanto a los criterios se deben seleccionar previamente para esto se recomienda tener en cuenta el valor de la importancia con el que cuenta cada una de los criterios por los que se debe medir para realizar el proceso de priorización de requisito.

Por otra parte se tiene en cuenta el principio básico de toma de decisiones ya que el sistema es multicriterio, multiexperto, multirequisito. Lo anteriormente planteado influye de forma directa en el sistema asegurando fiabilidad e interpretabilidad para realizar el proceso de priorización.

2.4.1 Principios para el diseño de interfaces.

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema. La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca a un sistema al éxito o al fracaso. Para el diseño de la interfaz se utilizaron diferentes principios que se enuncian a continuación:

Anticipación: La aplicación debe intentar anticiparse a las necesidades del usuario y no esperar a que el usuario tenga que buscar la información, recopilarla o invocar las herramientas que va a utilizar (47).

Autonomía: La computadora, la interfaz de usuario y el entorno de trabajo deben estar a disposición del usuario. Se debe dar al usuario un ambiente flexible para que pueda aprender rápidamente a usarla aplicación (47).

Ley de Fitt: El tiempo para alcanzar un objetivo es una función de la distancia y tamaño del objetivo. Es por ello, que es conveniente usar objetos grandes para las funciones importantes (47).

Objetos de Interfaz Humana: Los objetos de interfaz humana no son necesariamente los objetos que se encuentran en los sistemas orientados a objetos. Estos pueden ser vistos, escuchados, tocados o percibidos de alguna forma. Además, estos objetos deberían ser entendibles, consistentes y estables (47).

2.4.2 Tratamiento de excepciones

El tratamiento de errores facilita el buen funcionamiento de la aplicación. Una excepción es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias por lo que cuando se produce un error debido a la entrada o sencillamente la manipulación incorrecta de algún dato por parte del usuario. Cuando esto sucede se muestra un mensaje de error que le permita al usuario percatarse de que la acción que realizó es incorrecta y pueda corregir su error. En la herramienta se controlan los errores usando las técnicas de mensajes de alerta.

2.5 Patrón arquitectónico

Un patrón arquitectónico define la estructura básica de una aplicación, provee un subconjunto de subsistemas predefinidos, incluyendo reglas, lineamientos para conectarlos, pautas para su organización y constituye una plantilla de construcción (48).

Entre las ventajas del uso de patrones, se pueden encontrar:

- Permiten la reutilización de soluciones arquitectónicas de calidad.
- Son de gran ayuda para controlar la complejidad de un diseño.
- Facilitan la documentación de diseños arquitectónicos.
- Proporcionan un vocabulario común que mejora la comunicación entre diseñadores.

2.5.1 Patrones arquitectónicos

Se propone el uso del patrón arquitectónico N-Capas para el desarrollo de la aplicación, ya que permitirá hacer la implementación de forma más organizada. Aquí aparecen dos conceptos importantes: las capas que se encargan de la distribución lógica de los componentes, y los niveles que se refieren a la colocación física de los recursos. La característica principal de este patrón es que cada capa oculta las capas inferiores de las siguientes superiores a esta (49).

Algunas de las ventajas de utilizar este patrón son (50):

- Mejora las posibilidades de mantenimiento, debido a que cada capa es independiente de la otra, los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo.
- Flexibilidad, como cada capa puede ser manejada y escalada de forma independiente, la flexibilidad se incrementa.
- Disponibilidad, las aplicaciones pueden aprovechar la arquitectura modular de los sistemas habilitados usando componentes que escalan fácilmente lo que incrementa la disponibilidad.

Capa de presentación:

Esta capa es la encargada de presentar el sistema al usuario, le comunica la información y captura la información en un mínimo de proceso. Se comunica únicamente con la capa de negocio. Además se encuentran las clases encargadas de generar las interfaces gráficas y realizar validaciones a los datos entrados por el usuario. Esta capa se evidencia en la clase MainWindows, InstExperto, ModCriterio, entre otras que permiten al usuario interactuar con el sistema.

Capa de negocio:

En esta capa se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados. Esta capa se evidencia en la clase Prioriza, MatrizOWAWA, Save_Load_XML, también se utilizan otras clases que tienen la responsabilidad de realizar la parte del negocio. .

2.6 Patrones de Diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que se pueden encontrar a la hora de desarrollar una aplicación. Los patrones que se emplean en la solución son parte del conjunto de patrones GRASP. La utilización de este tipo de patrones permitió refinar el diseño y asignar las responsabilidades de las distintas clases, haciéndolas más sencillas, reutilizables y encapsuladas. A continuación se explican los patrones utilizados en nuestra aplicación.

Alta Cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, lo cual expresa que la información que almacena una clase debe de ser coherente y estar en la mayor medida de lo posible relacionada con la clase. En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta

cohesión tiene responsabilidades estrechamente relacionadas y poco complejas (51). Por ejemplo la clase Prioriza posee los métodos necesarios para satisfacer sus necesidades, sin sobrecargarla con métodos que no tengan que ver directamente con ella.

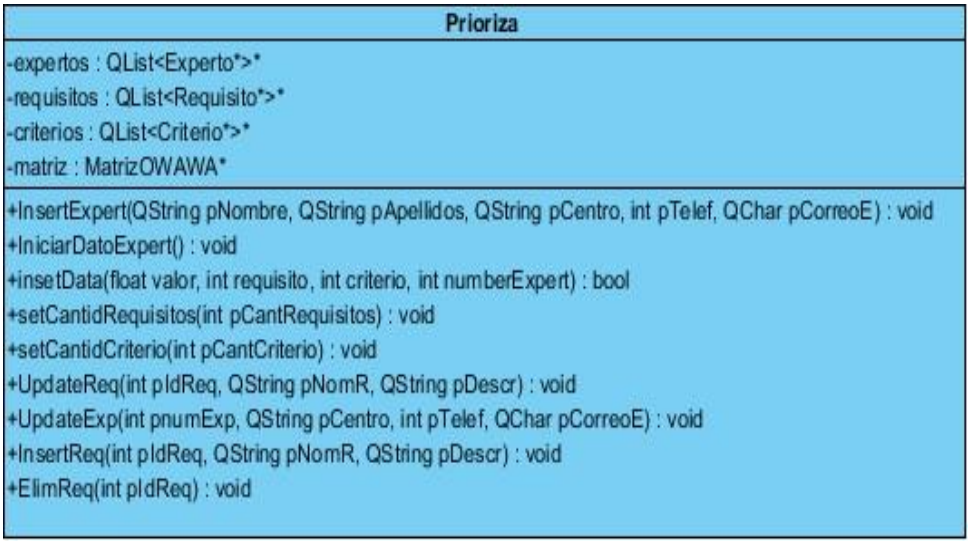


Figura 2. Ejemplo del patrón alta cohesión.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización (51). Por ejemplo la clase Prioriza necesita tener una instancia de la clase Requisito para realizas sus funciones.

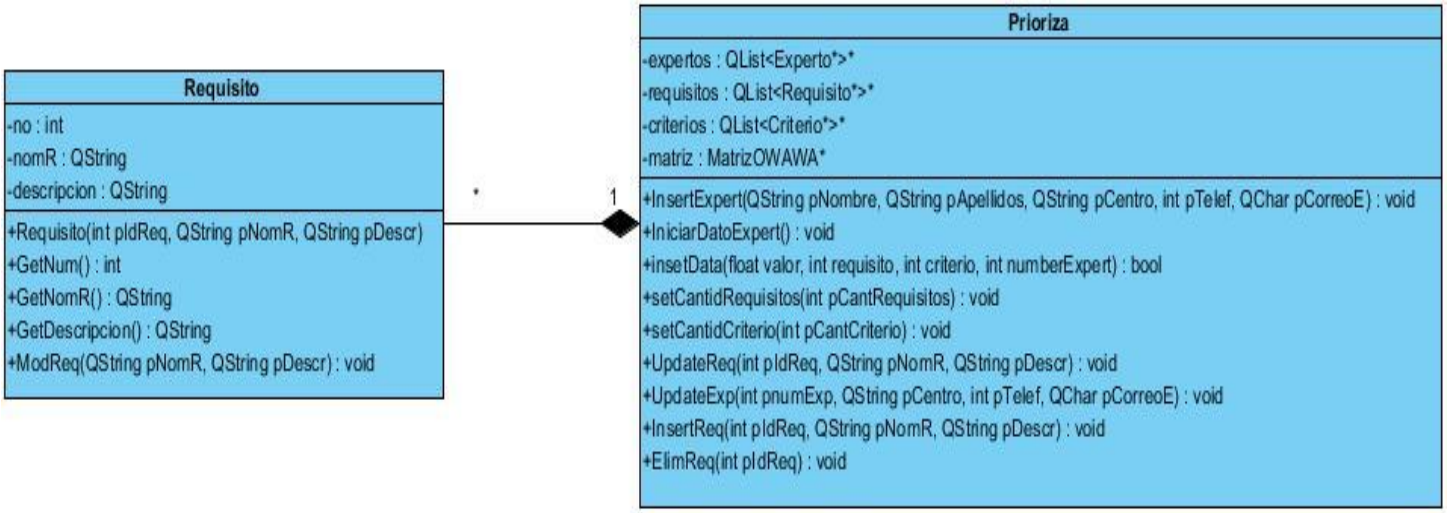


Figura 3. Ejemplo del patrón creador.

Controlador: Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto es para aumentar la reutilización de código y a la vez tener un mayor control. Recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento(51). Por ejemplo la clase Prioriza es la encargada de manejar los eventos de la clase Requisitos.

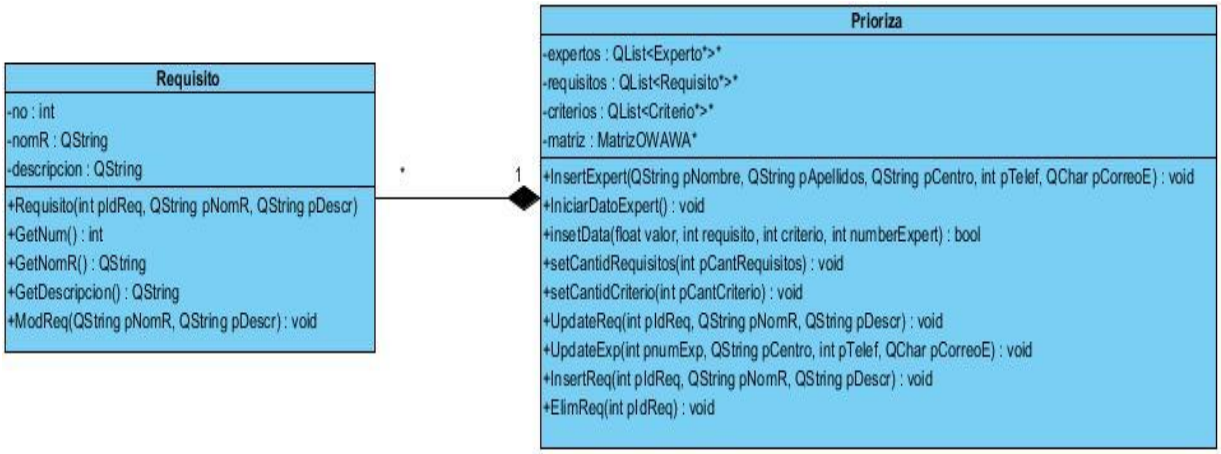


Figura 4. Ejemplo del patrón controlador

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está relacionada a otras clases. Una clase con bajo o débil acoplamiento no depende de muchas otras. Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. La solución es asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible, sin comprometer la funcionalidad por supuesto (51). Por ejemplo la clase Criterio solo depende de la clase Prioriza para realizar sus funciones.

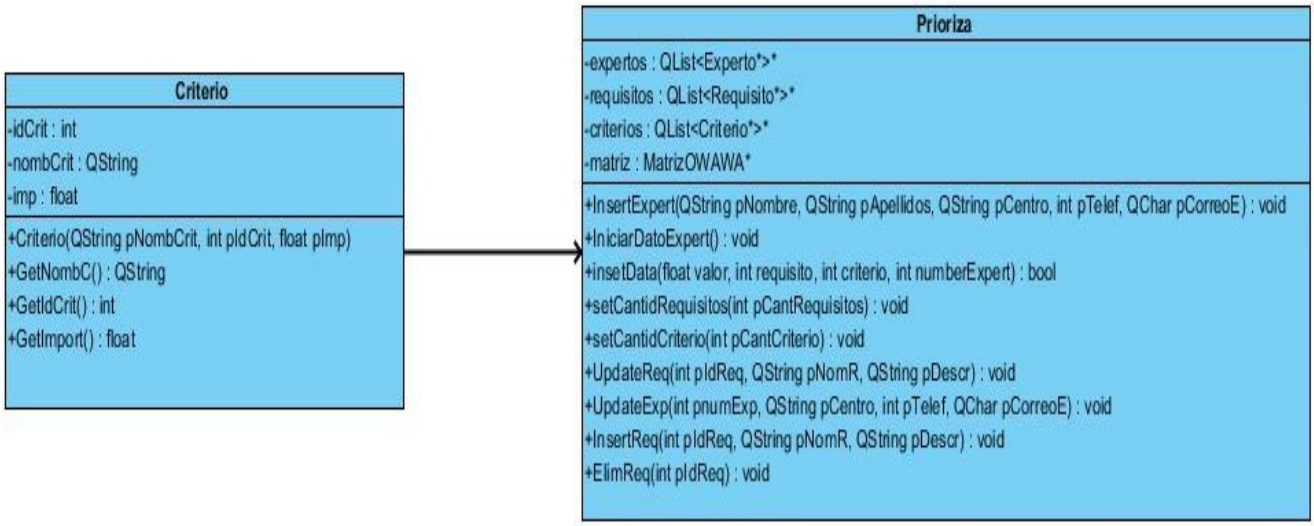


Figura 5. Ejemplo del patrón bajo acoplamiento

Experto: Un experto es una clase que tiene toda la información necesaria para implementar una responsabilidad. La respuesta es asignar la responsabilidad a la clase que contenga la información necesaria para cumplir la responsabilidad, o sea, la clase debe ser la experta en la información. La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos)(51). Por ejemplo la clase Prioriza tiene la información necesaria para obtener los datos de la clase Criterio.

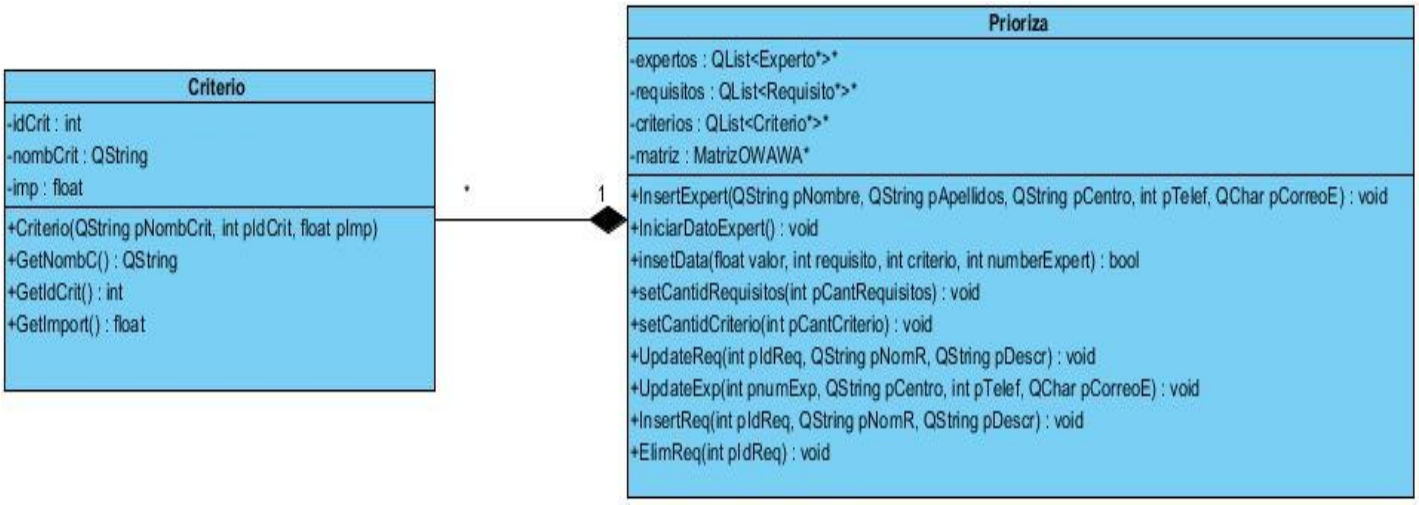


Figura 6. Ejemplo del patrón experto

2.7 Diseño de la solución propuesta

La metodología ágil XP plantea que la implementación del software debe realizarse de forma iterativa. Al culminar cada iteración se debe obtener un producto funcional, el mismo es probado por el cliente y de esta manera incrementar la visión de los desarrolladores con la opinión de este. Lo anteriormente expuesto contribuye a cumplir con la idea de potenciar las relaciones interpersonales como clave para lograr el éxito en el desarrollo del software en cuestión.

En el diseño de las aplicaciones realizadas bajo las reglas de la metodología XP, no es requerida la representación del sistema mediante diagramas de clase utilizando notación UML, sino que se utilizan otras técnicas, como las llamadas tarjetas CRC (Contenido, Responsabilidad y Colaboración). Sin embargo la utilización de estos diagramas puede aplicarse siempre y cuando influya en el mejoramiento de la comunicación entre el equipo de desarrollo y se enfoquen en la información importante (52).

2.7.1 Tarjetas CRC

Las tarjetas CRC son una metodología para el diseño de software orientado por objetos. Estas tienen como característica permitir romper con el modo de procedimiento y de pensamiento para apreciar mejor la tecnología de objetos. Además permiten que todo el equipo pueda contribuir al diseño del proyecto (53).

Tabla 18.Tarjeta CRC Prioriza

Tarjeta CRC	
Clase Prioriza	
Responsabilidades	**Colaboradores**

Esta es la clase que se encarga de controlar todo el sistema.
public:
void InsertExpert(QString pNombre,QString pApellidos, QString pCentro,int pTelef,QChar pCorreoE);
void IniciarDatoExpert();
bool insetData(float valor, int requisito, int criterio, int numberExpert);
void setCantidRequisitos (int pCantRequisitos){matriz-> setCantidRequisitos(pCantRequisitos);}
void setCantidCriterio (int pCantCriterio){matriz-> setCantidCriterio(pCantCriterio);}
void UpdateReq(int pIdReq, QString pNomR, QString pDescr);
void UpdateExp(int pnumExp, QString pCentro, int pTelef, QChar pCorreoE);
void InsertReq(int pIdReq, QString pNomR, QString pDescr);
void ElimReq(int pIdReq);
private:
QList<Experto*>* expertos;
QList<Requisito*>* requisitos;
QList<Criterio*>* criterios;
MatrizOWAWA* matriz;
	Clase experto.h
	Clase requisito.h
	Clase criterio.h
	Clase matrizowawa.h

Tabla 19. Tarjeta CRC MatrizOWAWA

Tarjeta CRC	
Clase MatrizOWAWA	
Responsabilidades	**Colaboradores**
Esta es la clase que se encarga de permitir el trabajo con la matriz.	Clase matrizowawa.h Clase experto.h Clase criterio.h

public:
MatrizOWAWA();
void setCantidRequisitos (int pCantRequisitos);
void setCantidCriterio();
bool insertExpert(float** data, int numberExpert);
bool insetDataExpert(float valor, int requisito, int criterio, int numberExpert);
bool createExpertNull(int numberExpert);
float** initMatrix(float** matrix, float initValor, int cantFilas, int cantColumnas);
bool redimencionar(int newCantRequisitos);
private:
QList<float**>* expertData;
int cantRequisitos;
int cantCriterios;
	Clase requisito.h

Tabla 20.Tarjeta CRC Requisito

Tarjeta CRC	
Clase Requisito	
Responsabilidades	**Colaboradores**

Esta es la clase que se encarga de realizar la gestión de los requisitos entrados al sistema.
public:
Requisito(int pIdReq, QString pNomR, QString pDescr);
int GetNum();
QString GetNomR();
QString GetDescripcion();
void ModReq(QString pNomR, QString Descr);
private:
int no;
QString nomR;
QString descripcion;
	Clase requisito.h

Tabla 21.Tarjeta CRC Experto

Tarjeta CRC	
Clase Experto	
Responsabilidades	**Colaboradores**

Esta es la clase que se encarga de realizar la gestión de los requisitos entrados al sistema.
public:
Experto(QString pNombre,QString pApellidos,int numExp,QString pCentro,int pTelef,QChar pCorreoE);
QString GetNombre();
QString GetApellido();
QString GetCentro();
int GetTelef();
int GetNumExp();
QChar GetCorreoE();
void ModExp(QString pCentro, int pTelef, QChar pCorreoE);
private:
QString nombre;
QString apellidos;
int numberExpert;
QString centro;
int telef;
QChar correoE;
	Clase experto.h

Tabla 22. Tarjeta CRC Criterio

Tarjeta CRC	
ClaseCriterio	
Responsabilidades	**Colaboradores**
Esta es la clase que se encarga de realizar el trabajo con los criterios seleccionado. **public:**	Clase criterio.h

Criterio(QString pNombCrit,int pIdCrit,float pImp);
QString GetNombC();
int GetIdCrit();
float GetImport();
 	
private:
QString nombCrit;
int idCrit;
float imp;	

Tabla 23. Tarjeta CRC MainWindows

Tarjeta CRC	
Clase MainWindows	
Responsabilidades	**Colaboradores**

En esta clase se realizará la implementación de la interfaz principal.
public:
Criterio(QString pNombCrit,int pIdCrit,float pImp);
QString GetNombC();
int GetIdCrit();
float GetImport();
 	
private:
QString nombCrit;
int idCrit;
float imp;
	Clase criterio.h

2.8 Conclusiones Parciales

En el presente capítulo se concluye que:

Se definieron los requisitos funcionales de la herramienta, permitiendo que se agruparan en 14 historias de usuarios.

Se realizó la estimación del tiempo para un total de 12 semanas, lo que permitió realizar el plan de iteraciones que define en qué momento se implementa cada requisito.

Se aplicó el patrón arquitectónico N-capas permitiendo definir la arquitectura de le herramienta en 2 capas: presentación y negocio.

.

Capítulo 3 Implementación y Prueba

El tercer capítulo y final se encargará de establecer los criterios para realizar la implementación y la validación del sistema desarrollado. Además se realizarán pruebas de aceptación al software para verificar el funcionamiento del mismo y de esa manera comprobar que todos los requisitos han sido correctamente implementados. Por otra parte se utiliza el método de caso de estudio para la validar la propuesta.

3.1 Estándares de codificación

Los estándares de codificación no son más que acciones que se toman para normalizar el código y este sea de fácil comprensión para los involucrados del proyecto. Además el seguir un estándar de programación te facilita como programador la modificación de tu propio código fuente aunque no estés trabajando en un equipo. Por lo general estos definen la forma en que van a ser declaradas las variables, las clases y los comentarios. Aunque en algunos estándares se incluyen los datos del programador, los cambios realizados al código fuente, entre otras especificaciones según el estándar a utilizar.

Se decidió escoger los estándares de codificación para el lenguaje de programación C++ definido por el centro VERTEX. Al usar dicho estándar se facilita la comprensión del código, además de cumplir con el objetivo de uniformidad.

3.1.1 Declaración de los espacios de nombres y las clases

Convenio de nombres

Se utilizará el idioma inglés para denotar el nombre de los archivos. De igual forma se utilizará este idioma para denotar todos los espacios de nombres, clases, métodos, variables constantes, y de manera general. En caso de no conocer la palabra correcta en idioma inglés se podrá utilizar, una palabra en español.

Espacio de nombres y de clases

Los indicadores para los espacios de nombres siempre serán sustantivos y comenzaran con letra inicial mayúscula, además el resto de la palabra se escribe en minúscula.

```
classPrioriza:publicQObject
```

Atributos de clases, parámetros, variables locales

Todos los caracteres se escribirán en minúscula. En caso de que existan nombre compuestos se utiliza la siguiente regla. El primer nombre se escribe con minúscula y a partir de ese momento cada nuevo nombre con letra inicial mayúscula.

```
Class MainWindow:public QMainWindow
{
Q_OBJECT
private:
QAction*exitAction;
QAction*open;
QAction*saveProyect;
QAction*myReq;
QAction*myCri;
QAction*myExp;
QAction*myMatriz;};
```

Directivas de pre-procesamiento

Son la única excepción para la cual se utilizarán *underscores* y mayúsculas. Se tratará siempre de prefijar los identificadores de las directivas para agruparlas lógicamente por función, área u otra dimensión.

```
#define PRIORIZA_H
```

Clases, espacios de nombre, implementación de métodos

Prioriza.h

```
Class Prioriza:public QObject
{
Q_OBJECT
```

```
private:
QList<Experto*>*expertos;
QList<Requisito*>*requisitos;
QList<Criterio*>*criterios;
MatrizOWAWA*matriz;
public:
explicit prioriza(QObject*parent=0);
prioriza();
void InsertExpert(QString pNombre, QString pApellidos, QString pCentro,int pTelef, QChar pCorreoE);
void IniciarDatoExpert();
bool insetData(float valor, int requisito, int criterio, int numberExpert);
void setCantidRequisitos(int pCantRequisitos){matriz->setCantidRequisitos(pCantRequisitos);}
void setCantidCriterio(int pCantCriterio){matriz->setCantidCriterio(pCantCriterio);}
void UpdateReq(int pIdReq,QString pNomR,QStringpDescr);
void UpdateExp(int pnumExp, QStringp Centro,int pTelef, QChar pCorreoE);
void InsertReq(int pIdReq,QString pNomR,QString pDescr);
void ElimReq(int pIdReq);
```

Prioriza.cpp

```
Void Prioriza::InsertReq(int pIdReq, QString pNomR, QString pDescr)
{
Int alien=pIdReq;
for(inti=0;i<requisitos->length();i++)
```

```
{
if(alien==requisitos->at(i)->GetNum())
{
this->tr("Existerequisito");
}
}
Requisito*rfn=new Requisito(pIdReq,pNomR,pDescr);
requisitos->append(rfn);
}
```

3.2 Implementación

En la fase de implementación, XP plantea la implementación de cada una de las HU. Al inicio se realiza un chequeo de cada una de las HU junto con el plan de iteraciones y se modifica en caso de ser necesario. Se crean tareas para ayudar a organizar la implementación de las HU asignando a un desarrollador la responsabilidad de su implementación. Estas tareas normalmente se escriben en un lenguaje técnico a diferencia de las HU que son escritas en el lenguaje del cliente. Para obtener el resultado esperado se decidió realizar la fase de implementación en 3 iteraciones. En el presente trabajo de diploma se expondrán dos ejemplos de cada iteración las restantes se encuentran en los anexos A2.

3.2.1 Iteración 1

En la primera iteración se garantizó la gestión de los datos de los expertos y los requisitos, creando las funciones necesarias que permiten dicha tarea.

Tabla 24. Historias de usuarios atendidas en la primera iteración.

Historias de Usuarios	Estimación(semanas)	Real (semanas)
Insertar requisito	2/3 semanas	2/3 semanas

Modificar requisito	2/3 semanas	2/3 semanas
Eliminar requisito	2/3 semanas	2/3 semanas
Insertar experto	2/3 semanas	2/3 semanas
Modificar experto	2/3 semanas	2/3 semanas
Eliminar experto	2/3 semanas	2/3 semanas

Tabla 25. Tarea para la HU#1

Tarea	
No de tarea: 1	**No de HU:** HU1
Nombre de la tarea: Implementar la funcionalidad insertar requisito	
Tipo de tarea: Desarrollo	**Estimación:** 2/3
Fecha de inicio: 20/2/2015	**Fecha fin:** 25/2/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Se realizan los métodos necesarios para que al insertar un requisito este cumpla con las restricciones mínimas para cumplir esta acción.	

Tabla 28. Tarea para la HU#6

Tarea	
No de tarea: 4	**No de HU:** HU6
Nombre de la tarea: Insertar experto	
Tipo de tarea: Desarrollo	**Estimación:** 2/3
Fecha de inicio: 2/3/2015	**Fecha fin:** 9/3/2015
Programador responsable: Alejandro Fuentes Barcelay	

Descripción: Se implementarán los métodos necesarios para que antes de ser insertado el experto, este deberá entrar los datos de manera correcta.

3.2.2 Iteración 2

Al realizar la segunda iteración se obtendrán tanto los criterios como la importancia de cada uno de ellos mediante los cuales se van a regir los expertos para realizar la priorización. Por otra parte se mostrará el resultado de la priorización y en que iteración debe implementarse.

Tabla 31.Historias de usuarios atendidas en la segunda iteración.

Historias de Usuario	Estimación (semanas)	Real (semanas)
Seleccionar Criterio	¾ semana	¾ semana
Seleccionar importancia de los criterios	2/3 semana	2/3 semana
Mostrar priorización	2 semana	2 semana
Mostrar el orden por requisitos	½ semana	½ semana
Seleccionar complejidad	½ semana	½ semana
Visualizar la matriz	½ semana	½ semana

Tabla 32. Tarea para la HU#4

Tarea	
No de tarea:7	**No de HU:** HU4
Nombre de la tarea: Implementar la funcionalidad seleccionar Criterio e importancia de los mismos	

Tipo de tarea: Desarrollo	**Estimación:** 1,5
Fecha de inicio: 25/3/2015	**Fecha fin:** 2/4/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Se creará la funcionalidad que permitirá al usuario seleccionar uno o varios criterios y a la vez decidir la importancia que debe tener dicho criterio.	

Tabla 34. Tarea para la HU#10

Tarea	
No de tarea:10	**No de HU:** HU10
Nombre de la tarea: Implementar la funcionalidad mostrar orden por requisitos	
Tipo de tarea: Desarrollo	**Estimación:** 2
Fecha de inicio: 5/3/2015	**Fecha fin:** 10/4/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Mediante esta tarea se implementarán los métodos necesarios para que, mediante la interfaz principal el usuario pueda visualizar el orden de los requisitos.	

3.2.3 Iteración 3

Para la tercera y última iteración se implementará la funcionalidad que permitirá la elaboración de los reportes, estos se deben obtener luego de realizar la priorización de requisitos.

Tabla 36. Historias de usuarios atendidas en la tercera iteración.

Historias de Usuarios	Estimación(semanas)	Real (semanas)
Guardar XML	1 semanas	1 semanas

Importar XML	1 semanas	1 semanas

Tabla 38. Tarea para la HU#11

Tarea	
No de tarea:11	**No de HU:** HU11
Nombre de la tarea: Guardar e importar XML	
Tipo de tarea: Desarrollo	**Estimación:** 2
Fecha de inicio: 30/4/2015	**Fecha fin:** 12/5/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Se crearán las funcionalidades que permitan al usuario guardar e importar un archivo XML con los datos insertados en la aplicación para su posterior uso.	

3.3 Pruebas

Unas de las vías más importantes para determinar el estado de la calidad de un producto de software es el proceso de pruebas. Estas están dirigidas a componentes del sistema en su totalidad, con el objetivo de medir el grado en que cumple con los requisitos. En ellas se usan casos de prueba, especificados de forma estructurada mediante técnicas. Sus objetivos, métodos y técnicas usadas se describen en el plan de prueba (54).

Las pruebas constituyen una actividad fundamental en los procesos de desarrollo de software. Estas permiten detectar la presencia de errores que pudieran generar las entradas o salidas de datos y comportamientos inapropiados durante su ejecución (54).

Al aplicarles las pruebas al software se deben seguir un conjunto de estrategias para lograr que estas se hagan en el menor tiempo posible y con la calidad requerida, además de garantizar que arrojen los resultados esperados (54).

La metodología XP plantea dos grupos de pruebas: las unitarias, que las realizan los programadores, estas pruebas son las encargadas de verificar el código de forma automática. El otro grupo de pruebas son las de aceptación, que están dedicadas a probar si al final de una iteración se obtuvieron las funcionalidades especificadas y además que den las respuesta como el usuario desea.

3.3.1 Pruebas de Aceptación

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de su puesta en marcha es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada "prueba de aceptación". En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique y eliminar la influencia de conflictos de intereses, y para que sea lo más objetiva posible, la prueba de aceptación no debería ser responsabilidad de los ingenieros de software que han desarrollado el producto (54).

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente (54). En el presente trabajo de diploma se seleccionaron los casos de pruebas más importantes y el resto fue ubicado en los anexos A3.

Tabla 39 Prueba de aceptación para la HU " Insertar requisito"

Caso de Prueba de Aceptación	
Código: HU_1_P1	**Historia de Usuario:** 1
Nombre: Insertar requisito	
Descripción: Prueba para la funcionalidad de Insertar requisito	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de insertar requisito.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción de insertar requisito • Introduce el nombre del requisito, identificador y la descripción • Se da clic en el botón aceptar	

Resultado Esperado: Que la herramienta permita insertar requisito. La herramienta debe mostrar en la escena los requisitos insertados.
Evaluación de la Prueba: Prueba satisfactoria

Tabla 42 Prueba de aceptación para la HU "Insertar experto"

Caso de Prueba de Aceptación	
Código: HU_6_P1	**Historia de Usuario:**6
Nombre: Insertar experto	
Descripción: Prueba para la funcionalidad de Insertar experto	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de insertar experto.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción de insertar experto • Se introducen los datos nombre, apellidos, correo electrónico, número de experto, teléfono, centro y años de experiencia. • Se da clic en el botón aceptar.	
• **Resultado Esperado**: Que la herramienta permita insertar experto. La herramienta debe mostrar una escena con los expertos insertados en el sistema.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla 45 Prueba de aceptación para la HU "Seleccionar Criterio"

Caso de Prueba de Aceptación	
Código: HU_4_P2	**Historia de Usuario:**4
Nombre: Seleccionar Criterio	
Descripción: Prueba para la funcionalidad de Seleccionar Criterio	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de seleccionar criterio.	

Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción seleccionar criterio • Se selecciona tanto el criterio y su importancia • Se da clic en el botón aceptar
• **Resultado Esperado**: Que la herramienta permita seleccionar criterio. Los criterios seleccionados deben estar presentes en la matriz.
Evaluación de la Prueba: Prueba satisfactoria

Tabla 47 Prueba de aceptación para la HU "Mostrar orden por requisito"

Caso de Prueba de Aceptación	
Código: HU_10_P2	**Historia de Usuario:**10
Nombre: Mostrar orden por requisito	
Descripción: Prueba para la funcionalidad de Mostrar orden por requisito	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de ver y dentro de esta la opción matriz.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción ver • Luego se selecciona la opción matriz • Luego se llenan los datos de la matriz • Luego se selecciona la opción priorizar • Muestra los requisitos ordenados	
• **Resultado Esperado**: Que la herramienta permita modificar experto.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla 48 Prueba de aceptación para la HU "Guardar XML"

Caso de Prueba de Aceptación	
Código: HU_11_P3	**Historia de Usuario:**11
Nombre: Guardar XML	
Descripción: Prueba para la funcionalidad Guardar XML	

Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de guardar los datos ya insertados en la misma.
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción guardar • Se escoge la dirección donde se va a guardar • Se selecciona la opción aceptar
• **Resultado Esperado**: Que la herramienta permita Guardar XML en una dirección específica.
Evaluación de la Prueba: Prueba satisfactoria

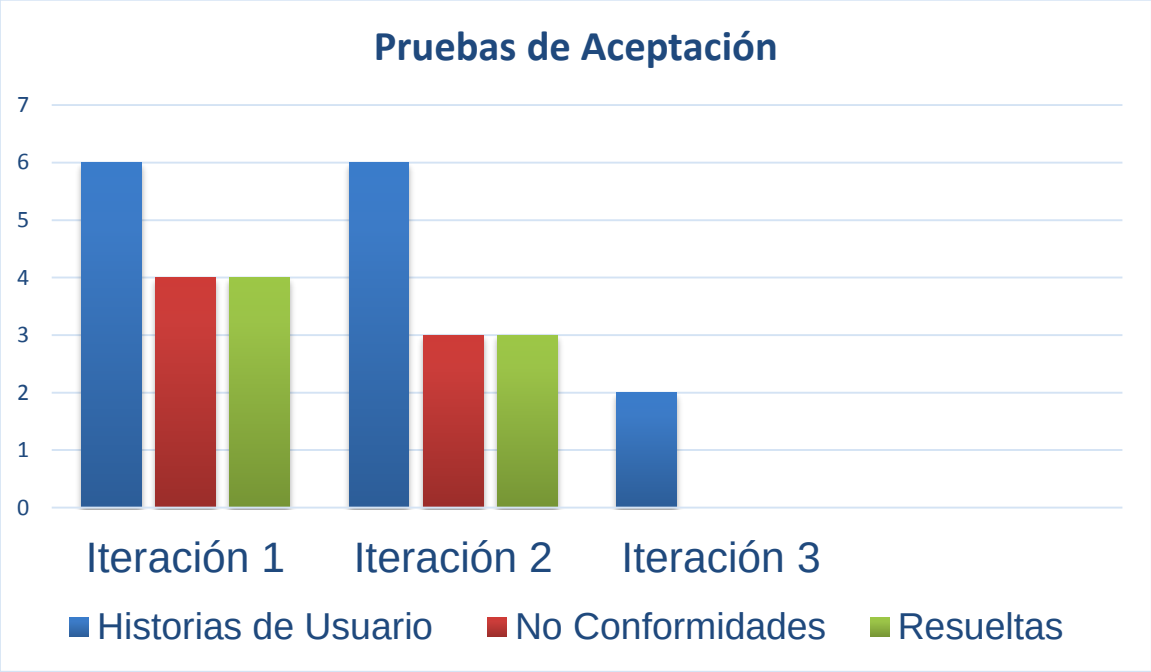


Figura 7: Validación de las Historias de Usuarios

Una vez aplicadas las pruebas, estas arrojaron los siguientes resultados:

- En la primera iteración se le realizaron pruebas de aceptación a 6 historias de usuario, detectando 4 no conformidades, las cuales fueron resueltas satisfactoriamente.
- Para la segunda iteración se atendieron 6 historias de usuario, donde se detectaron 3 no conformidades, las cuales fueron resultas satisfactoriamente.
- Mientras que en la última iteración no se detectaron no conformidades, por lo que la herramienta se encuentra sin errores y lista para su uso.

3.4 Caso de estudio: Rehabilitador de Marcha

En el Centro Entorno Interactivo 3D (VERTEX) se está desarrollando un sistema informático Rehabilitador de Marcha que permite proyectar imágenes sobre una cinta rodante, para la rehabilitación de los pacientes con discapacidades motoras en extremidades inferiores. El sistema guía de forma visual a los pacientes con problemas neuromotores en su proceso de recuperación y les indica cómo debe dar los pasos sobre la estera transportadora.

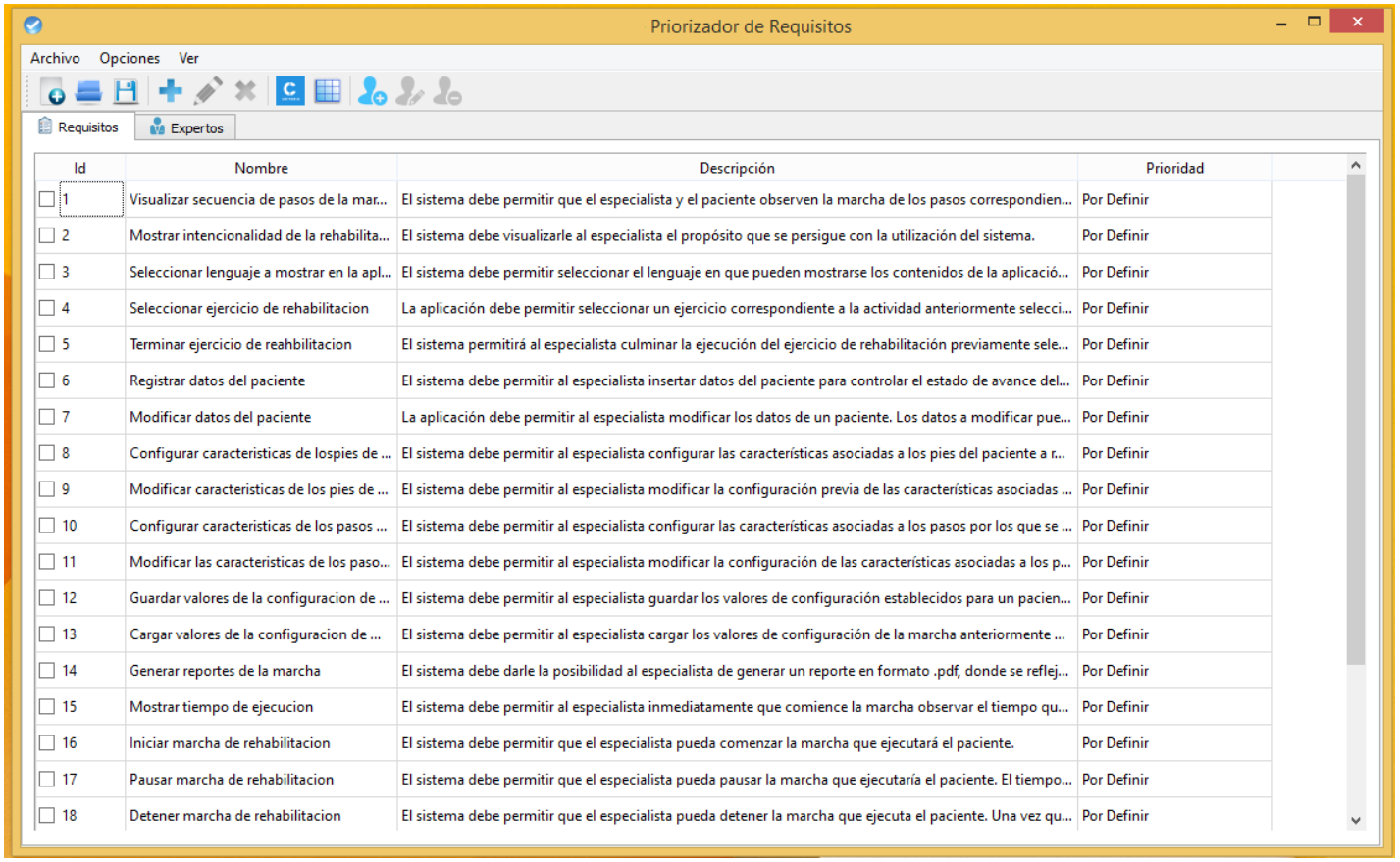
El rehabilitador le brinda información a los especialistas, acerca de las características adecuadas con las que debe tratar al paciente para proceder con su rehabilitación. Constituye además una solución que se adapta de acuerdo a las especificidades de cada paciente, y un medio eficaz donde los especialistas valoran la recuperación del paciente.

La solución permite el registro de pacientes en las terapias, el control de la evolución que posea el paciente a medida que se rehabilita; y genera reportes en PDF de los resultados de un paciente que realiza varias sesiones de rehabilitación.

Perfil Salud

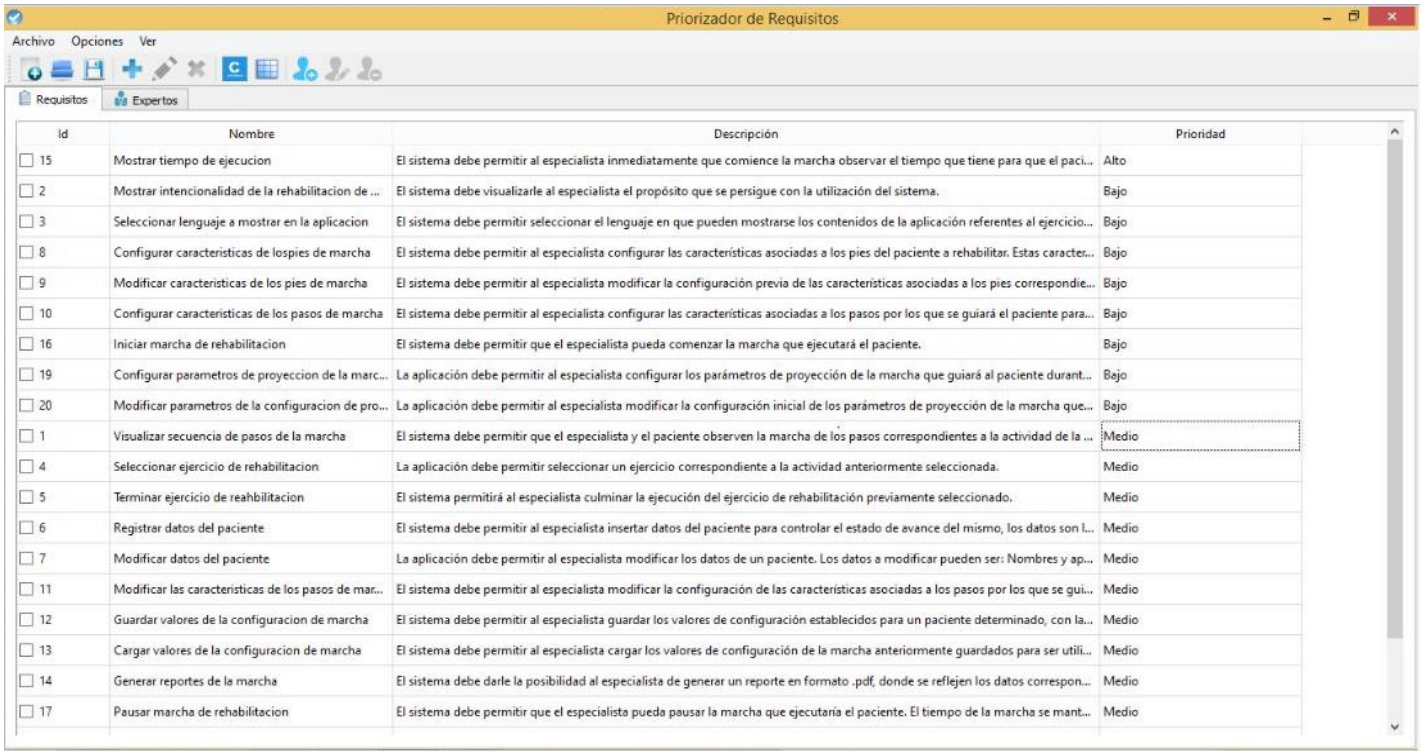
- Permite que las sesiones de rehabilitación realizadas por los pacientes queden reflejadas en un reporte o guía, para la posterior evaluación de los especialistas.
- Permite guiar visualmente a los pacientes con respecto a la distancia, cadencia, posición y ángulo en el cual tiene que dar los pasos.
- Permite la configuración manual por el doctor de los parámetros de la marcha de rehabilitación según el paciente que sea atendido.
- Permite almacenar los perfiles de los pacientes según sus padecimientos y continuar el proceso de rehabilitación en diferentes secciones.

El rehabilitador cuenta con requisitos 22 requisitos. A continuación se muestran las pantallas que explican de forma visual como se realiza el proceso de PR en la herramienta.



Id	Nombre	Descripción	Prioridad
1	Visualizar secuencia de pasos de la mar...	El sistema debe permitir que el especialista y el paciente observen la marcha de los pasos correspondien...	Por Definir
2	Mostrar intencionalidad de la rehabilita...	El sistema debe visualizarle al especialista el propósito que se persigue con la utilización del sistema.	Por Definir
3	Seleccionar lenguaje a mostrar en la apl...	El sistema debe permitir seleccionar el lenguaje en que pueden mostrarse los contenidos de la aplicació...	Por Definir
4	Seleccionar ejercicio de rehabilitacion	La aplicación debe permitir seleccionar un ejercicio correspondiente a la actividad anteriormente selecci...	Por Definir
5	Terminar ejercicio de reahbilitacion	El sistema permitirá al especialista culminar la ejecución del ejercicio de rehabilitación previamente sele...	Por Definir
6	Registrar datos del paciente	El sistema debe permitir al especialista insertar datos del paciente para controlar el estado de avance del...	Por Definir
7	Modificar datos del paciente	La aplicación debe permitir al especialista modificar los datos de un paciente. Los datos a modificar pue...	Por Definir
8	Configurar caracteristicas de lospies de ...	El sistema debe permitir al especialista configurar las características asociadas a los pies del paciente a r...	Por Definir
9	Modificar caracteristicas de los pies de ...	El sistema debe permitir al especialista modificar la configuración previa de las características asociadas ...	Por Definir
10	Configurar caracteristicas de los pasos ...	El sistema debe permitir al especialista configurar las características asociadas a los pasos por los que se ...	Por Definir
11	Modificar las caracteristicas de los paso...	El sistema debe permitir al especialista modificar la configuración de las características asociadas a los p...	Por Definir
12	Guardar valores de la configuracion de ...	El sistema debe permitir al especialista guardar los valores de configuración establecidos para un pacien...	Por Definir
13	Cargar valores de la configuracion de ...	El sistema debe permitir al especialista cargar los valores de configuración de la marcha anteriormente ...	Por Definir
14	Generar reportes de la marcha	El sistema debe darle la posibilidad al especialista de generar un reporte en formato .pdf, donde se reflej...	Por Definir
15	Mostrar tiempo de ejecucion	El sistema debe permitir al especialista inmediatamente que comience la marcha observar el tiempo qu...	Por Definir
16	Iniciar marcha de rehabilitacion	El sistema debe permitir que el especialista pueda comenzar la marcha que ejecutará el paciente.	Por Definir
17	Pausar marcha de rehabilitacion	El sistema debe permitir que el especialista pueda pausar la marcha que ejecutaría el paciente. El tiempo...	Por Definir
18	Detener marcha de rehabilitacion	El sistema debe permitir que el especialista pueda detener la marcha que ejecuta el paciente. Una vez qu...	Por Definir

Figura 8.Interfaz principal



Id	Nombre	Descripción	Prioridad
15	Mostrar tiempo de ejecucion	El sistema debe permitir al especialista inmediatamente que comience la marcha observar el tiempo que tiene para que el paci...	Alto
2	Mostrar intencionalidad de la rehabilitacion de ...	El sistema debe visualizarle al especialista el propósito que se persigue con la utilización del sistema.	Bajo
3	Seleccionar lenguaje a mostrar en la aplicacion	El sistema debe permitir seleccionar el lenguaje en que pueden mostrarse los contenidos de la aplicación referentes al ejercicio...	Bajo
8	Configurar caracteristicas de lospies de marcha	El sistema debe permitir al especialista configurar las características asociadas a los pies del paciente a rehabilitar. Estas caracter...	Bajo
9	Modificar caracteristicas de los pies de marcha	El sistema debe permitir al especialista modificar la configuración previa de las características asociadas a los pies correspondie...	Bajo
10	Configurar caracteristicas de los pasos de marcha	El sistema debe permitir al especialista configurar las características asociadas a los pasos por los que se guiará el paciente para...	Bajo
16	Iniciar marcha de rehabilitacion	El sistema debe permitir que el especialista pueda comenzar la marcha que ejecutará el paciente.	Bajo
19	Configurar parametros de proyeccion de la marc...	La aplicación debe permitir al especialista configurar los parámetros de proyección de la marcha que guiará al paciente durant...	Bajo
20	Modificar parametros de la configuracion de pro...	La aplicación debe permitir al especialista modificar la configuración inicial de los parámetros de proyección de la marcha que...	Bajo
1	Visualizar secuencia de pasos de la marcha	El sistema debe permitir que el especialista y el paciente observen la marcha de los pasos correspondientes a la actividad de la ...	Medio
4	Seleccionar ejercicio de rehabilitacion	La aplicación debe permitir seleccionar un ejercicio correspondiente a la actividad anteriormente seleccionada.	Medio
5	Terminar ejercicio de reahbilitacion	El sistema permitirá al especialista culminar la ejecución del ejercicio de rehabilitación previamente seleccionado.	Medio
6	Registrar datos del paciente	El sistema debe permitir al especialista insertar datos del paciente para controlar el estado de avance del mismo, los datos son l...	Medio
7	Modificar datos del paciente	La aplicación debe permitir al especialista modificar los datos de un paciente. Los datos a modificar pueden ser: Nombres y ap...	Medio
11	Modificar las caracteristicas de los pasos de mar...	El sistema debe permitir al especialista modificar la configuración de las características asociadas a los pasos por los que se gui...	Medio
12	Guardar valores de la configuracion de marcha	El sistema debe permitir al especialista guardar los valores de configuración establecidos para un paciente determinado, con la...	Medio
13	Cargar valores de la configuracion de marcha	El sistema debe permitir al especialista cargar los valores de configuración de la marcha anteriormente guardados para ser utili...	Medio
14	Generar reportes de la marcha	El sistema debe darle la posibilidad al especialista de generar un reporte en formato .pdf, donde se reflejen los datos correspon...	Medio
17	Pausar marcha de rehabilitacion	El sistema debe permitir que el especialista pueda pausar la marcha que ejecutaría el paciente. El tiempo de la marcha se mant...	Medio

Figura 9. Interfaz principal

Una vez insertado los requisitos y realizado el proceso de priorización por parte de la herramienta, se decidió comparar el proceso real contra el proceso de la herramienta, coincidiendo la priorización de la herramienta con el orden de implementación de los requisitos del proyecto.

3.5 Conclusiones parciales

En este capítulo se realizó la implementación y validación de la herramienta propuesta que permitió obtener una primera versión de la aplicación para ser utilizada en el proceso de priorización de requisitos de software.

Se desarrollaron las fases de construcción y prueba de la metodología de desarrollo XP, que posibilitó seleccionar las pruebas de aceptación para validar el funcionamiento de las interfaces de usuario.

Se realizaron tres iteraciones en la etapa de construcción de la aplicación, así como la ejecución de tareas de programación correspondiente a cada historia de usuario, permitiendo organizar el proceso de desarrollo de manera adecuada.

Se profundizó en el estudio de las pruebas unitarias y de aceptación, seleccionando las últimas como las más indicadas para comprobar el funcionamiento del software, debido a que demuestran la satisfacción del cliente con el producto.

Se utilizó el método de caso de estudio para validar la propuesta desarrollada, arrojando excelente resultados.

Conclusiones Generales

Luego de culminada la investigación se concluye:

- Se realizó un estudio de la priorización de requisitos resumiendo que los métodos de priorización de requisitos analizados no son factibles y pueden ser engorrosos cuando existe una lista larga de requisitos,
- Se seleccionó el modelo lingüístico 2-tuplas ya que este permite aumentar la interpretación de los modelos de CW, proporcionando resultados lingüísticos que sean fácilmente interpretables.
- Se desarrolló un sistema para la priorización de requisitos a través de criterios definidos lo que posibilitará un mejor control del tiempo y recursos de los proyectos productivos de software.
- Se validó el sistema con las pruebas de aceptación, y se resolvieron todas las no conformidades detectadas; por lo que se demuestra que la herramienta para la priorización de requisitos basada en la computación con palabras está lista para su uso y permite la correcta planificación del cronograma de los proyectos en el desarrollo de software.
- Se probó la herramienta haciendo uso del método caso de estudio arrojando excelente resultados y siendo a la vez una guía para los involucrados del proyecto a la hora de decidir el orden de implementación de los requisitos.

Recomendaciones

Se recomienda que:

- La herramienta de priorización de requisitos se utilice en la Universidad de las Ciencias Informáticas para contribuir al adecuado proceso de planificación de software de los proyectos productivos.
- La implementación de una funcionalidad que permita graficar la priorización de requisitos a través de la traslación simbólica para aumentar la usabilidad del software, ya que el usuario tendrá la posibilidad de visualizar cómo se comportan los requisitos en dicha gráfica.

Referencias bibliográficas

1. WIEGERS, Karl. First Things First: Prioritising Requirements. Published in Software Development Magazine. . 1999.

2. YOUNG, Ralph. The Requirements Enginneering Handbook. . 2004.
3. ZADEH, L. A. *The concept of a linguistic variable and its applications to approximate reasoning Information Sciences.* [no date].
4. 12. BERTHOLD, M. H. DAVID Intelligent Data Analysis: An Introduction Springe. . 2010.
5. PÉREZ, K, L., MAIKEL and ESPINILLA, MACARENA. A linguistic software requirement prioritization model with heterogeneous information. . 2012.
6. CARLOS MARTÍN AZZOLINI. *Un Enfoque de Priorizaci ón de Requerimientos, a partir de la Segmentación de las Preferencias de los Stakeholders*. 2011.
7. ESPINILLA, M.R., DA, LIU, JUN and MARTÍNEZ, Luis. A heterogeneous evaluation model for assessing sustainable energy: A Belgian case study.IEEE World Congress on Computational Intelligence. . 2010. P. 1–8.
8. SOMMERVILLE, I. Requirement Engineering. Addison-Wesley. Ed. 5ta. England, Wokingham. . 1996.
9. FIRESMITH, D. Prioritizing Requirements. . 2004. P. 35–47.
10. JHONNY GÓMEZ CHALACÁN. *VALIDACIÓN Y APLICACIÓN DE TÉCNICAS DE PRIORIZACIÓN DE REQUISITOS* [online]. CALI - COLOMBIA : UNIVERSIDAD DE SAN BUENAVENTURA, 2012. Available from: http://bibliotecadigital.usbcali.edu.co/jspui/bitstream/10819/1332/1/Validaci%C3%B3n_Aplicaci%C3%B3n_T%C3%A9cnicas_G%C3%B3mez_2012.pdf autor: JHONNY GÓMEZ CHALACÁN
11. MA Q. *The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements: A Systematic Literature Review. Auckland University of Technology* [online]. Auckland : University of Technology, 2009. Available from: http://aut.researchgateway.ac.nz/bitstream/handle/10292/833/MaQ.pdf?sequence=3
12. HATTON, S. *Choosing the right prioritization method. 19 th Australian Conference on Software Engineering.* [online]. 2008. Available from: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4483241&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4483241
13. BERANDER, P. *Prioritization of Stakeholder Need in Software Engineering* [online]. 2004. Available from: http://www.bth.se/fou/forskinfo.nsf/0/0c0a10b9ebb6019ec1256fef00556599/$file/Patrik%20Berander%20-%20Licentiate%20Thesis.pdf
14. • LEFFINGWELL, D and WIDRING, D. Managing Software Requirements – A unified approach. Addison-Wesley. Upper Saddle River, Nj. . 2000.
15. REGNELL, B, HOST, M, NATT OCH DAG, J, BEREMARK, P and HJELM, T. *An Industrial case study on distributed prioritization in market –driven Requirements Engineering for Packaged Software.* 2001.
16. SAATY, T, L. *Axiomatic foundations of the analytic hierarchy process*. 1986.
17. SAATY, T, L. How to make decision: The Analytic Hierarchy Process". European Journal of Operational Research. (48). North Holland. Pg: 9-26. . 1990. P. 9–26.
18. SAATY, T, L. Decision-Making whit the AHP: Why is the principal eigenvector necessary ?". ISAHP 2001, Berne, Switzerland, August 2-4. *University of Pittsburgh*. 2001. P. 2–4.
19. KUMAR, S and VAIDYAA, O. Analytic hierarchy process: An overview of applications. *European*

Journal of Operational Research. 2006. Vol. 169, no. 1, p. 1–29.
20. SHILIANG, W and SHIBO, W. • "Application of AHP Method into synthetic evaluation of venture capital investment projects". Enterprise Economy. . 2004. P. 72–75.
21. RODRÍGUEZ BELLO, S. *Toma de decisión Multicriterio con AHP, ANP y LD.* Colombia : Universidad Nacional de Colombia, 2007.
22. KARLSSON, J. Software Requirements Prioritizing. *IEEE Software*. 1996.
23. WILTOLD PEDRYCZ, ROBERTA PARREIRAS and PETR EKEL. FUZZY MULTICRITERIA DECISION-MAKING MODELS, METHODS AND APPLICATIONS. . 2011.
24. MARIANA IVETH CHAIREZ. Toma de decisiones. .
25. FRANCISCO HERRERA and LUIS MARTÍNEZ. A 2-Tuple Fuzzy Linguistic Representation Model for Computing with Words. . 2000.
26. MENDEL, J. Z., L, TRILLAS, E, YAGER, R, LAWRY, J, HAGRAS, H and GUADARRAMA, S. *What computing with words means to me IEEE* . 2010.
27. GRANVILLE, KIM and KOHOUT. Granular relational computing with semiotic descriptors using BK-products of fuzzy relations, Computing with words. . 2001. P. 89–146.
28. THIELE. *On semantic models for investigating Computing with Words* [online]. Abr 1998. [Accessed 14 April 2015]. Available from: https://eldorado.tu-dortmund.de/bitstream/2003/5329/2/ci3298_doc.pdf
29. RODRÍGUEZ DOMÍNGUEZ, Rosa Ma. *Un Nuevo Modelo para Procesos de Computación con Palabras en Toma de Decisión Lingüística.. Junio de 2010*. 2010.
30. BONISSONE and DECKER. Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-Off Precision and Complexity. [online]. 27 March 2013. Available from: http://arxiv.org/abs/1304.3425
31. YAGER R.R. *Non-numeric multi-criteria multi-person decision making.Group Decision and Negotation.* 1993.
32. HERRERA, Francisco and MARTÍNEZ, Luis. A model based on linguistic 2-tuples for dealing with multigranularity hierarchical linguistic contexts in multiexpert decision-making. IEEE Transactions on Systems, Man and Cybernetics. Part B: Cybernetics,. . 2001. P. 227–234.
33. ZULUETA VÉLIZ, Yeleny. *Modelos de evaluación de la importancia del impacto ambiental en contextos complejos bajo incertidumbre*. 2014.
34. HERRERA, Francisco, MARTÍNEZ, Luis and SÁNCHEZ, P. J. F. Herrera, L. Martínez, and P.J. Sánchez. Managing non-homogeneous information in group decision making. European Journal of OperationalResearch. .
35. TORRA, V. N and YASUO. Modeling decisions. Information Fusion and Aggregation Operators. Springer. . 2007.
36. XU, Z. *Dependent OWA operators Modeling Decisions for Artificial Intelligence.* 2006.
37. BOONGOEN, T. and S., QIANG. Clus-DOWA: A new dependent OWA operator. *IEEE World Congress on Computational Intelligence* . 2008. P. 1057–1063.
38. *Que es la progración extrema?* [online]. 2015. Available from: http://www.codejobs.biz /es/blog/2013/01/09/que-es-la-programacion-extrema-xp#sthash.3eejKTnn.M0Cc7WCJ.dpbs
39. Tutor de C++. [online]. 10 May 2015. [Accessed 12 May 2015]. Available from:

http://decsai.ugr.es/~jfv/ed1/c++/cdrom3/TIC-CD/web/portada/faqs/faqs1_3.htm
40. Qt Quick. [online]. May 2015. [Accessed 12 June 2015]. Available from: http://www.qt.io/Product/Qt-Framework/IDE--TOOL/
41. Extensible Markup Language (XML). [online]. [Accessed 28 May 2015]. Available from: http://www.w3.org/XML/
42. Requerimientos Funcionales y No Funcionales (RF/RNF). [online]. [Accessed 12 April 2015]. Available from: http://ingenieriadesoftware.bligoo.com.mx/requerimientos-funcionales-y-no-funcionales-rf-rnf
43. FASES DE LA PROGRAMCIÓN EXTREMA. Fases. [online]. [Accessed 14 May 2015]. Available from: http://programacionextrema.tripod.com/fases.htm
44. EMILIO A SÁNCHEZ, PATRICIO LETELIER and JOSÉ H. CANÓS. Mejorando la gestión de historias de usuario en eXtreme Programming. . 16 April 2015.
45. Ciclo de vida de un proyecto XP. [online]. [Accessed 12 April 2015]. Available from: http://oness.sourceforge.net/proyecto/html/ch05s02.html
46. LiderDeProyecto.com / Estimación de Esfuerzo del Proyecto. [online]. [Accessed 19 April 2015]. Available from: http://www.liderdeproyecto.com/manual/estimacion_de_esfuerzo_del_proyecto.html
47. Diseño de Interfaces de Usuario. [online]. [Accessed 24 April 2015]. Available from: http://www.monografias.com/trabajos10/diusuar/diusuar.shtml
48. LARMAN, Craig. *UML y PATRONES Una introducción al análisis y diseño orientado a objetos y al procesounificado.* 2014.
49. REYNOSO, Carlos and KICILLOF, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. UNIVERSIDAD DE BUENOS AIRES. . 2004. P. pág. 24.
50. ARQUITECTURA EN CAPAS. [online]. [Accessed 16 April 2015]. Available from: http://arquitecturaencapas.blogspot.com/2011/08/arquitectura-3-capas%20programacion-por.html
51. DANYA RAO. GRASP Design Principles. [online]. 15 April 2015. Available from: http://www.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/rao.pdf.
52. JOSKOWICZ, José. Reglas y Prácticas en eXtreme Programming. [online]. 2 October 2008. Available from: http://iie.fing.edu.uy/~josej/docs/XP-Jose Joskowicz.pdf
53. ZAMBRANO, Omar. Qué son y para qué sirven las tarjetas CRCs. . 16 2014.
54. MARTÍNEZ and SALAZAR, Eduardo. Informática Jurídica. [online]. 1 April 2014. Available from: http://www.informaticajuridica.com/trabajos/Propuesta_Procedimiento_para_realizar_pruebas_Caja_Blanca_aplicaciones_desarrol lan_lenguaje_Python.asp.

Bibliografía

1. WIEGERS, Karl. First Things First: Prioritising Requirements. Published in Software Development Magazine. . 1999.

2. YOUNG, Ralph. The Requirements Enginneering Handbook. . 2004

3. *Especificación de requisitos de software.* 15 April 2015.

4. HERNÁNDEZ SAMPIERI, C. ROBERTO, CARLOS FERNÁNDEZ Y LUCIO, COLLADO and LUCIO, PILA BAPTISTA. *Metodología de la Investigación.* México : Editorial Mc Graw Hill, 2010.

5. CARLOS MARTÍN AZZOLINI. *Un Enfoque de Priorizaci ón de Requerimientos, a partir de la Segmentación de las Preferencias de los Stakeholders*. 2011.

6. ESPINILLA, M.R., DA, LIU, JUN and MARTÍNEZ, Luis. A heterogeneous evaluation model for assessing sustainable energy: A Belgian case study.IEEE World Congress on Computational Intelligence. . 2010. P. 1–8.

7. SOMMERVILLE, I. Requirement Engineering. Addison-Wesley. Ed. 5ta. England, Wokingham. . 1996.

8. FIRESMITH, D. Prioritizing Requirements. . 2004. P. 35–47.

9. JHONNY GÓMEZ CHALACÁN. *VALIDACIÓN Y APLICACIÓN DE TÉCNICAS DE PRIORIZACIÓN DE REQUISITOS* [online]. CALI - COLOMBIA : UNIVERSIDAD DE SAN BUENAVENTURA, 2012. Available from: http://bibliotecadigital.usbcali.edu.co/jspui/bitstream/10819/1332/1/Validaci%C3%B3n_Aplicaci%C3%B3n_T%C3%A9cnicas_G%C3%B3mez_2012.pdf autor: JHONNY GÓMEZ CHALACÁN

10. MA Q. *The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements: A Systematic Literature Review. Auckland University of Technology* [online]. Auckland : University of Technology, 2009. Available from: http://aut.researchgateway.ac.nz/bitstream/handle/10292/833/MaQ.pdf?sequence=3

11. HATTON, S. *Choosing the right prioritization method. 19 th Australian Conference on Software Engineering.* [online]. 2008. Available from: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4483241&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4483241

12. BERANDER, P. *Prioritization of Stakeholder Need in Software Engineering* [online]. 2004. Available from: http://www.bth.se/fou/forskinfo.nsf/0/0c0a10b9ebb6019ec1256fef00556599/$file/Patrik%20Berander%20-%20Licentiate%20Thesis.pdf

13. • LEFFINGWELL, D and WIDRING, D. Managing Software Requirements – A unified approach. Addison-Wesley. Upper Saddle River, Nj. . 2000.

14. REGNELL, B, HOST, M, NATT OCH DAG, J, BEREMARK, P and HJELM, T. *An Industrial case study on distributed prioritization in market –driven Requirements Engineering for Packaged Software.* 2001.

15. SAATY, T, L. *Axiomatic foundations of the analytic hierarchy process*. 1986.

16. SAATY, T, L. How to make decision: The Analytic Hierarchy Process". European Journal of Operational Research. (48). North Holland. Pg: 9-26. . 1990. P. 9–26.

17. SAATY, T, L. Decision-Making whit the AHP: Why is the principal eigenvector necessary ?". ISAHP 2001, Berne, Switzerland, August 2-4. *University of Pittsburgh*. 2001. P. 2–4.

18. KUMAR, S and VAIDYAA, O. Analytic hierarchy process: An overview of applications. *European Journal of Operational Research*. 2006. Vol. 169, no. 1, p. 1–29.

19. SHILIANG, W and SHIBO, W. • "Application of AHP Method into synthetic evaluation of venture capital investment projects". Enterprise Economy. . 2004. P. 72–75.

20. RODRÍGUEZ BELLO, S. *Toma de decisión Multicriterio con AHP, ANP y LD.* Colombia : Universidad Nacional de Colombia, 2007.

21. KARLSSON, J. Software Requirements Prioritizing. *IEEE Software*. 1996.

22. WILTOLD PEDRYCZ, ROBERTA PARREIRAS and PETR EKEL. FUZZY MULTICRITERIA DECISION-MAKING MODELS, METHODS AND APPLICATIONS. . 2011.

23. MARIANA IVETH CHAIREZ. Toma de decisiones. .

24. FRANCISCO HERRERA and LUIS MARTÍNEZ. A 2-Tuple Fuzzy Linguistic Representation Model for Computing with Words. . 2000.

25. MENDEL, J. Z., L, TRILLAS, E, YAGER, R, LAWRY, J, HAGRAS, H and GUADARRAMA, S. *What computing with words means to me IEEE .* 2010.

26. GRANVILLE, KIM and KOHOUT. Granular relational computing with semiotic descriptors using BK-products of fuzzy relations, Computing with words. . 2001. P. 89–146.

27. THIELE. *On semantic models for investigating Computing with Words* [online]. Abr 1998. [Accessed 14 April 2015]. Available from: https://eldorado.tu-dortmund.de/bitstream/2003/5329/2/ci3298_doc.pdf

28. RODRÍGUEZ DOMÍNGUEZ, Rosa Ma. *Un Nuevo Modelo para Procesos de Computación con Palabras en Toma de Decisión Lingüística.. Junio de 2010*. 2010.

29. BONISSONE and DECKER. Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-Off Precision and Complexity. [online]. 27 March 2013. Available from: http://arxiv.org/abs/1304.3425

30. YAGER R.R. *Non-numeric multi-criteria multi-person decision making.Group Decision and Negotation.* 1993.

31. HERRERA, Francisco and MARTÍNEZ, Luis. A model based on linguistic 2-tuples for dealing with multigranularity hierarchical linguistic contexts in multiexpert decision-making. IEEE Transactions on Systems, Man and Cybernetics. Part B: Cybernetics,. . 2001. P. 227–234.

32. ZULUETA VÉLIZ, Yeleny. *Modelos de evaluación de la importancia del impacto ambiental en contextos complejos bajo incertidumbre*. 2014.

33. HERRERA, Francisco, MARTÍNEZ, Luis and SÁNCHEZ, P. J. F. Herrera, L. Martínez, and P.J. Sánchez. Managing non-homogeneous information in group decision making. European Journal of OperationalResearch. .

34. TORRA, V. N and YASUO. Modeling decisions. Information Fusion and Aggregation Operators. Springer. . 2007.

35. XU, Z. *Dependent OWA operators Modeling Decisions for Artificial Intelligence.* 2006.

36. BOONGOEN, T. and S., QIANG. Clus-DOWA: A new dependent OWA operator. *IEEE World Congress on Computational Intelligence .* 2008. P. 1057–1063.

37. *Que es la progración extrema?* [online]. 2015. Available from: http://www.codejobs.biz /es/blog/2013/01/09/que-es-la-programacion-extrema-xp#sthash.3eejKTnn.M0Cc7WCJ.dpbs

38. Tutor de C++. [online]. 10 May 2015. [Accessed 12 May 2015]. Available from: http://decsai.ugr.es/~jfv/ed1/c++/cdrom3/TIC-CD/web/portada/faqs/faqs1_3.htm

39. Qt Quick. [online]. May 2015. [Accessed 12 June 2015]. Available from: http://www.qt.io/Product/Qt-Framework/IDE--TOOL/

40. Extensible Markup Language (XML). [online]. [Accessed 28 May 2015]. Available from: http://www.w3.org/XML/

41. Requerimientos Funcionales y No Funcionales (RF/RNF). [online]. [Accessed 12 April 2015]. Available from: http://ingenieriadesoftware.bligoo.com.mx/requerimientos-funcionales-y-no-funcionales-rf-rnf

42. FASES DE LA PROGRAMCIÓN EXTREMA. Fases. [online]. [Accessed 14 May 2015]. Available from: http://programacionextrema.tripod.com/fases.htm

43. EMILIO A SÁNCHEZ, PATRICIO LETELIER and JOSÉ H. CANÓS. Mejorando la gestión de historias de usuario en eXtreme Programming. . 16 April 2015.

44. Ciclo de vida de un proyecto XP. [online]. [Accessed 12 April 2015]. Available from: http://oness.sourceforge.net/proyecto/html/ch05s02.html

45. LiderDeProyecto.com / Estimación de Esfuerzo del Proyecto. [online]. [Accessed 19 April 2015]. Available from: http://www.liderdeproyecto.com/manual/estimacion_de_esfuerzo_del_proyecto.html

46. Diseño de Interfaces de Usuario. [online]. [Accessed 24 April 2015]. Available from: http://www.monografias.com/trabajos10/diusuar/diusuar.shtml

47. LARMAN, Craig. *UML y PATRONES Una introducción al análisis y diseño orientado a objetos y al procesounificado.* 2014.

48. REYNOSO, Carlos and KICILLOF, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. UNIVERSIDAD DE BUENOS AIRES. . 2004. P. pág. 24.

49. ARQUITECTURA EN CAPAS. [online]. [Accessed 16 April 2015]. Available from: http://arquitecturaencapas.blogspot.com/2011/08/arquitectura-3-capas%20programacion-por.html

50. DANYA RAO. GRASP Design Principles. [online]. 15 April 2015. Available from: http://www.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/rao.pdf.

51. JOSKOWICZ, José. Reglas y Prácticas en eXtreme Programming. [online]. 2 October 2008. Available from: http://iie.fing.edu.uy/~josej/docs/XP-Jose Joskowicz.pdf

52. ZAMBRANO, Omar. Qué son y para qué sirven las tarjetas CRCs. . 16 2014.

53. MARTÍNEZ and SALAZAR, Eduardo. Informática Jurídica. [online]. 1 April 2014. Available from: http://www.informaticajuridica.com/trabajos/Propuesta_Procedimiento_para_realizar_pruebas_Caja_Blanca_aplicaciones_desarrol lan_lenguaje_Python.asp.

Anexo

Anexo A1. Historias de usuario

Tabla 2. Historia de Usuario 2

Historia de Usuario	
Número : 2	Nombre de HU : Modificar requisito
Fecha : 6-03-2015	Usuario : Cliente
Prioridad en el Negocio : Alta	Riesgo de Desarrollo : Alto
Puntos Estimados : 2/3	Iteración Asignada :1
Descripción: El sistema debe permitir al usuario modificar los requisitos que desee priorizar. Esta acción se debe realizar luego de dar clic sobre el requisito que se quiere modificar o buscando la opción que permite realizar dicha tarea.	
Observaciones:	

Tabla 3. Historia de usuario 3

Historia de Usuario	
Número : 3	Nombre de HU : Eliminar requisito
Fecha : 6-03-2015 Prioridad en el Negocio : Alta	Usuario : Cliente Riesgo de Desarrollo : Alto

Puntos Estimados : 2/3	Iteración Asignada :1
Descripción: El sistema debe permitir al usuario eliminar los requisitos que desee no priorizar para esta acción seleccionar el requisito o los requisitos que se desean eliminar. Para realizar esto se debe marcar los requisitos o el requisito que se desea eliminar.	
Observaciones:	

Tabla 5. Historia de usuario 5

Historia de Usuario	
Número: 5	Nombre de HU: Seleccionar importancia de los criterios
Fecha: 7-03-2015	Usuario: Cliente
Prioridad en el Negocio: Alta	Riesgo de Desarrollo: Alto
Puntos Estimados: 2/3	Iteración Asignada: 2
Descripción: El sistema debe permitir al usuario seleccionar el valor de la importancia de los criterios por los cuales se llevará a cabo la priorización. En este caso se debe seleccionar un número entre 1 y 5.	
Observaciones:	

Tabla 7. Historia de usuario 7

Historia de Usuario	
Número: 7	Nombre de HU: Modificar experto
Fecha: 6-03-2015 Prioridad en el Negocio: Alta Puntos Estimados: 2/3	Usuario: Cliente Riesgo de Desarrollo: Alto Iteración Asignada:1
Descripción: El sistema debe permitir al usuario modificar los datos de los expertos. Esta acción se debe realizar marcando al experto que se desea modificar. Luego para realizar dicha tarea se debe dar clic sobre el experto o buscar la opción modificar experto existente en el sistema.	
Observaciones:	

Tabla 8. Historia de usuario 8

Historia de Usuario	
Número: 8	Nombre de HU: Eliminar experto
Fecha: 6-03-2015 Prioridad en el Negocio: Alta	Usuario: Cliente Riesgo de Desarrollo: Alto
Puntos Estimados: 2/3	Iteración Asignada: 1
Descripción: El sistema debe permitir al usuario eliminar a los expertos que este no estime necesarios en el proceso de priorización. . Para realizar esto	

se debe marcar los expertos o el experto que se desea eliminar y dar a la opción eliminar experto del sistema.
Observaciones:

Tabla 9. Historia de usuario 9

Historia de Usuario	
Número: 9	Nombre de HU: Mostrar priorización
Fecha: 10-03-2015	Usuario: Cliente
Prioridad en el Negocio: Alta	Riesgo de Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada: 2
Descripción: El sistema debe permitir al usuario visualizar al usuario el resultado de la priorización. Luego de insertados los datos en la matriz y dar clic en el botón priorizar, en la interfaz principal se llenará la columna Prioridad donde se encontrará la complejidad de cada requisito.	
Observaciones:	

Tabla 12. Historia de usuario 12

Historia de Usuario	
Número: 12	Nombre de HU: Importar XML
Fecha: 10-03-2015	Usuario: Cliente

Prioridad en el Negocio: Media	Riesgo de Desarrollo: Medio
Puntos Estimados: 1	Iteración Asignada: 3
Descripción: El sistema debe crear un archivo en XML mediante el cual se guardarán los datos de los expertos, los requisitos a priorizar y los criterios seleccionados para realizar la priorización. Observaciones:	

Tabla 11. Historia de usuario 11

Historia de Usuario	
Número: 11	Nombre de HU: Guardar XML
Fecha: 10-03-2015	Usuario: Cliente
Prioridad en el Negocio: Media	Riesgo de Desarrollo: Medio
Puntos Estimados: 1	Iteración Asignada: 3
Descripción: El sistema debe crear un archivo en XML mediante el cual se guardarán los datos de los expertos, los requisitos a priorizar y los criterios seleccionados para realizar la priorización.	
Observaciones:	

Tabla 13. Historia de usuario 13

Historia de Usuario

Número: 13	Nombre de HU: Seleccionar complejidad
Fecha: 10-03-2015	Usuario: Cliente
Prioridad en el Negocio: Alta	Riesgo de Desarrollo: Alto
Puntos Estimados: 1/2	Iteración Asignada: 2
Descripción: El sistema debe permitir al experto escoger la complejidad de los requisitos según el criterio entrado. La acción de seleccionar clasificación se realizará solamente cuando este cargada la matriz cuando todos los datos son entrados correctamente al sistema.	
Observaciones:	

Tabla 14. Historia de usuario 14

Historia de Usuario	
Número: 14	Nombre de HU: Visualizar matriz
Fecha: 10-03-2015	Usuario: Cliente
Prioridad en el Negocio: Alta	Riesgo de Desarrollo: Alto
Puntos Estimados: 1/2	Iteración Asignada: 3
Descripción: El sistema debe permitir al usuario luego de introducir los datos de forma correcta a la aplicación visualizar la matriz, para después insertar las clasificaciones de los requisitos y realizar la priorización.	
Observaciones:	

Anexo A2 Tareas de ingeniería

Tabla 26. Tarea para la HU#2

Tarea	
No de tarea: 2	**No de HU:** HU 2
Nombre de la tarea: Modificar requisito	
Tipo de tarea: Desarrollo	**Estimación:** 2/3
Fecha de inicio: 26/2/2015	**Fecha fin:** 2/3/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Para que esta funcionalidad se realicé primero se verificará si los datos introducidos por el usuario son correctos y luego ejecutar la modificación de los datos en el sistema.	

Tabla 27. Tarea para la HU#3

Tarea	
No de tarea: 3	**No de HU:** HU3
Nombre de la tarea: Eliminar requisito	
Tipo de tarea: Desarrollo	**Estimación:** 2/3
Fecha de inicio: 3/3/2015	**Fecha fin:** 6/3/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Para esta funcionalidad se implementaron los métodos necesarios para eliminar uno o varios requisitos a la vez.	

Tabla 29. Tarea para la HU#7

Tarea

No de tarea: 5	**No de HU:** HU7
Nombre de la tarea: Modificar experto	
Tipo de tarea: Desarrollo	**Estimación:** 2/3
Fecha de inicio: 17/3/2015	**Fecha fin:** 20/3/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Se creará la funcionalidad que permitirá al desarrollador modificar un experto luego de ser seleccionado en la interfaz de los requisitos.	

Tabla 30. Tarea para la HU#8

Tarea	
No de tarea: 6	**No de HU:** HU8
Nombre de la tarea: Eliminar experto	
Tipo de tarea: Desarrollo	**Estimación:** 2/3
Fecha de inicio: 23/3/2015	**Fecha fin:** 27/3/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Se desarrollarán los métodos que permita eliminar ya sea a un experto como a varios expertos luego de ser seleccionados en la interfaz de los expertos.	

Tabla 33. Tarea para la HU#9

Tarea	
No de tarea:9	**No de HU:** HU9
Nombre de la tarea: Mostrar priorización	
Tipo de tarea: Desarrollo	**Estimación: 3**
Fecha de inicio: 10/3/2015	**Fecha fin:** 2/4/2015

Programador responsable: Alejandro Fuentes Barcelay
Descripción: Se creará la funcionalidad que permitirá al desarrollador visualizar la priorización luego de haberse introducido correctamente los datos en la matriz.

Tabla 36. Tarea para la HU#13, HU#14

Tarea	
No de tarea:9	**No de HU:** HU13
Nombre de la tarea: Implementar la selección de lo complejidad en la matriz de priorización	
Tipo de tarea: Desarrollo	**Estimación:** ½
Fecha de inicio: 31/3/2015	**Fecha fin:** 2/4/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Se creará la funcionalidad que permitirá al desarrollador seleccionar la complejidad de los requisitos, basándose en los criterios seleccionados.	

Tabla 37. Tarea para la HU#14

Tarea	
No de tarea:9	**No de HU:** HU14
Nombre de la tarea: Implementar la matriz de priorización y su visualización	
Tipo de tarea: Desarrollo	**Estimación:** ½
Fecha de inicio: 31/3/2015	**Fecha fin:** 2/4/2015
Programador responsable: Alejandro Fuentes Barcelay	
Descripción: Se implementarán los métodos necesarios para realizar la matriz y la visualización de la misma mediante una interfaz de usuario.	

Anexo A3 Casos de pruebas

Tabla 40 Prueba de aceptación para la HU " Modificar requisito"

Caso de Prueba de Aceptación	
Código: HU_2_P1	**Historia de Usuario:** 2
Nombre: Modificar requisito	
Descripción: Prueba para la funcionalidad de Modificar requisito	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de modificar requisito.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción de modificar requisito • Selecciona el requisito que se va a modificar • Se introducen los datos a modificar • Se da clic en el botón modificar	
Resultado Esperado: Que la herramienta permita modificar requisito.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla 41 Prueba de aceptación para la HU "Eliminar requisito"

Caso de Prueba de Aceptación	
Código: HU_3_P1	**Historia de Usuario:**3
Nombre: Eliminar requisito	
Descripción: Prueba para la funcionalidad de Eliminar requisito	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de eliminar requisito.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción de eliminar requisito • Se introduce el número identificador del requisito que se desea eliminar. • Se da clic en el botón eliminar	

Resultado Esperado: Que la herramienta permita eliminar el requisito. La herramienta debe mostrar una escena donde se compruebe que realmente fue eliminado el requisito.
Evaluación de la Prueba: Prueba satisfactoria

Tabla 43 Prueba de aceptación para la HU "Modificar experto"

Caso de Prueba de Aceptación	
Código: HU_7_P1	**Historia de Usuario:**7
Nombre: Modificar experto	
Descripción: Prueba para la funcionalidad de Modificar experto	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de modificar experto.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción de modificar experto • Selecciona el experto que se va a modificar • Se introducen los datos a modificar • Se da clic en el botón modificar	
• **Resultado Esperado**: Que la herramienta permita modificar experto.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla 44 Prueba de aceptación para la HU "Eliminar experto"

Caso de Prueba de Aceptación	
Código: HU_8_P1	**Historia de Usuario:**8
Nombre: Eliminar experto	
Descripción: Prueba para la funcionalidad de Eliminar experto	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de eliminar experto.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción de eliminar experto	

• Se introduce el número del experto que se desea eliminar. • Se da clic en el botón eliminar
Resultado Esperado: Que la herramienta permita modificar requisito. La herramienta debe mostrar una escena donde se compruebe que realmente fue eliminado el experto.
Evaluación de la Prueba: Prueba satisfactoria

Tabla 46 Prueba de aceptación para la HU "Mostrar priorización"

Caso de Prueba de Aceptación	
Código: HU_9_P2	**Historia de Usuario:**9
Nombre: Mostrar priorización	
Descripción: Prueba para la funcionalidad de Mostrar priorización	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de ver y dentro de esta la opción matriz.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción ver • Luego se selecciona la opción matriz • Luego se llenan los datos de la matriz • Luego se selecciona la opción priorizar	
• **Resultado Esperado**: Que la herramienta permita mostrar la priorización de requisito. Esta opción solo dará los números obtenidos luego de ejecutar el modelo de 2-tuplas.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla 49 Prueba de aceptación para la HU "Importar XML"

Caso de Prueba de Aceptación	
Código: HU_12_P3	**Historia de Usuario:**12
Nombre : Importar XML	

Descripción: Prueba para la funcionalidad Importar XML
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de importar un XML.
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción abrir • Se escoge la dirección donde se encuentra el archivo XML • Se selecciona la opción aceptar
• **Resultado Esperado**: Que la herramienta permita Importar XML de una dirección específica.
Evaluación de la Prueba: Prueba satisfactoria

Tabla 50 Prueba de aceptación para la HU "Importar XML"

Caso de Prueba de Aceptación	
Código: HU_11_P3	**Historia de Usuario:**13
Nombre: Seleccionar complejidad	
Descripción: Prueba para la funcionalidad Seleccionar complejidad	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de guardar los datos ya insertados en la misma.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción insertar expertos • Se selecciona la opción insertar requisitos • Se selecciona la opción los criterios con su correspondiente valor de importancia • Se selecciona la opción Ver y dentro de esta la opción matriz o dar clic sobre el icono representativo de la matriz en la interfaz principal. • Seleccionar en cada celda de la matriz la complejidad del requisito con respecto a los criterios seleccionados.	
• **Resultado Esperado**: Que la herramienta permita Guardar XML en una dirección específica.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla 51 Prueba de aceptación para la HU "Importar XML"

Caso de Prueba de Aceptación	
Código: HU_11_P3	**Historia de Usuario:**14
Nombre: Visualizar matriz de priorización	
Descripción: Prueba para la funcionalidad Visualizar matriz de priorización	
Condiciones de Ejecución: La aplicación debe de estar en ejecución y debe dar la opción de guardar los datos ya insertados en la misma.	
Entrada/Pasos de Ejecución: • Se ejecuta la aplicación • Se selecciona la opción insertar expertos • Se selecciona la opción insertar requisitos • Se selecciona la opción los criterios con su correspondiente valor de importancia • Se selecciona la opción Ver y dentro de esta la opción matriz o dar clic sobre el icono representativo de la matriz en la interfaz principal.	
• **Resultado Esperado**: Que la herramienta permita visualizar la matriz pero sin las complejidades insertadas.	
Evaluación de la Prueba: Prueba satisfactoria	

Glosario de Términos

C++: es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos.

XP: Programación Extrema.

CW: Computación con palabras.

HU: Historias de usuario

VERTEX: Centro de Entornos Iterativo 3D.

XML: Lenguaje de marcas extensible.

HTML: *HyperText Markup Language* («lenguaje de marcas de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones.

IDE: Entorno de Desarrollo Integrado.