Universidad de las Ciencias Informáticas

Facultad 3

Título: Herramienta de apoyo a la toma de decisiones para el proceso de priorización de requisitos de software.

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

Autoras: *Tania Pérez Ramírez.*

Malena Pereda Lemus.

Tutores: *Ing. Yoandris Espinosa Núñez.*

Msc. José Ramón Ruíz de Zarate y del Cueto.

La Habana, Cuba

Junio de 2013

"Que el maquillaje no apague tu risa, Que el equipaje no lastres tus alas, Que el calendario no venga con prisa, Que el diccionario detenga las balas, Que las persianas corrijan la aurora, Que gane el quiero la guerra del puedo, Que los que esperan no cuenten las horas, Que los que matan se mueran de miedo. Que el fin del mundo te pille bailando, Que el escenario nos tiñen las canas , Que nunca sepas ni cómo ni cuándo ni ciento volando ni ayer ni mañana… "

Joaquín Sabina.

Declaración de autoría

Declaramos ser las únicas autoras del Trabajo de Diploma que lleva por Título: “Herramienta de apoyo a la toma de decisiones para el proceso de priorización de requisitos de software”. Le otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Tania Pérez Ramírez Malena Pereda Lemus

_____ _____

Ing. Yoandris Espinosa Núñez Msc. José Ramón Ruíz de Zárate y del Cueto

Datos de contacto

Datos de los tutores:

Ing. Yoandris Espinosa Núñez

Correo: yoandris@uci.cu

La Habana, Cuba.

Msc. José Ramón Ruíz de Zárate y del Cueto

Correo: zarate@uci.cu

La Habana, Cuba.

Agradecimientos

Muchas son las personas a las que solo agradecer no sería suficiente. Ha sido una verdadera prueba de fuego con disímiles obstáculos, imposibles de romper sin una mano amiga a la que agarrarse o sin palabras o frases que te iluminen el camino a recorrer. No son el fin de 5 años, sí no la de una etapa de vida que muere para que surja otra con nuevas oportunidades y magníficas expectativas.

A mi papi lindo gracias y mil veces gracias por darme la oportunidad de existir, por enseñarme el arte de vivir, por apoyarme en todas mis travesías a veces muy locas y aunque casi nunca estuvieses de acuerdo. Te quiero, te adoro, te amo infinitamente. A mi mami que nunca estamos de acuerdo pero la vida sería muy aburrida si fuera diferente no crees?. Te amo y te querré aunque no te lo diga. A mi tía Griselda Murias, la española, una de las personas de las que siempre busco para nutrirme de fuerza y voluntad. A mis hermanas que siempre han querido controlarme pero sé que me quieren como soy. A mis primas Yiliam por guiarme en estos últimos momentos y preocuparse, a primona Kenia que la distancia no nos impide estar cerca. A mi linda sobrina Lily, mi luz, mi motor impulsor. A mis sobrinas kamy e isa que a pesar de la ausencia y la distancia existe un amor infinito y tierno. A mi abuela que sin su ayuda esta tesis no hubiese salido, un beso. A mis primos, mis tías y mis tíos, en fin esa gran familia a la que pertenezco.

Y cómo no voy a agradecer a esa familia de amigos que puedo darme el lujo de lucir, con los que he compartido tantos momentos imborrables. Creo que lo que mucha gente ve como nuestra dificultad para mi es nuestra mayor virtud: la diferencia de caracteres. A Yanerkis por su carácter, a Yanet por su sinceridad, a Yaneisys por simpatía, a Yoce por su comprensión, A Nolvis por sus locuras. A Gisel, Analiet, Joicel, en fin a mis grandes amigos de la UCI. No por ser los últimos amigos en llegar en esta etapa son los menos reconocidos, mi piquetón de la alianza francesa, son lo máximo , los quiero Jessi, Mary , Osvaldo, Suci, Irina, Ani, Willy, Raiza , Lourdes, Reinier Julian en fin merci beacoup. Y por supuesto a la mejor profe de francés que pueda tener Daylén.

Y a ti que te conocí pero las distancias, el tiempo, las circunstancias no nos han dejado ser feliz o simplemente te voy a conocer ahora pero que voy a disfrutar cada proceso en el que estaré a tu lado aunque tenga que dar pasos con no estén en el mapa porque al final sé que serás por siempre y para siempre mi petit prince y yo tu flor de notre B612. vous serez toujours dans mon coeur, l'histoire d'amour la plus belle que j'ai eu et j'aurai. Je t'aime.

Tanita

Dedicatoria

A mi papi, por ser mi guía, mi gurú. A mis sobrinas mis razones de ser, mis hermosas mariposas .

Resumen

Cuando las expectativas del cliente son altas, los plazos son cortos, y los recursos son limitados, en un proyecto de desarrollo de software, se debe asegurar que el producto informático contenga las funciones más esenciales. Establecer en cada paso la prioridad relativa de estos, permite una construcción secuencial del sistema para proporcionar un elevado valor del producto. La Priorización de Requisitos de Software constituye un proceso de toma de decisiones perteneciente a la fase de análisis de la Ingeniería de Requisitos. Decidir entre un conjunto de requisitos atendiendo a varios aspectos como costo, importancia, entre otros, puede convertirse en un proceso complejo. Sin embargo no siempre se cuenta con las técnicas o herramientas para llevar a cabo dicho proceso, o no siempre se emplea la técnica que realmente necesitan los stakeholders según las características particulares del proceso de negociación. Existen una gran cantidad de técnicas, métodos, procesos, para priorizar los requisitos por lo que escoger entre ellas se convierte en un proceso complejo y lleno de incertidumbres.

El presente trabajo propone una herramienta de apoyo a la toma de decisiones que contiene un conjunto de técnicas, métodos, procesos con procedimientos heterogéneos para establecer prioridad entre los requisitos de software. La selección de estas técnicas, métodos y procesos se basó además de sus características particulares, en la calidad de los resultados que arrojan, y en las facilidades de uso para posibilitarle al usuario escoger entre un diapasón de opciones y así realizar el proceso de priorización con la que más se ajuste a sus necesidades y expectativas.

Palabras clave:

requisitos, priorización, toma de decisiones, herramienta de apoyo

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Introducción

En las últimas décadas, el proceso de desarrollo del software es un punto de gran interés para la comunidad internacional. Cuba, no está enajenada de este desarrollo, cuenta con empresas de software como Desoft, Transoft, SoftCaribe, Albet S.A, esta última comercializadora de productos y servicios informáticos que son desarrollados en la Universidad de las Ciencias Informáticas (UCI). La UCI juega un importante rol en el proyecto de informatización del país como forjadora de profesionales calificados y como productora de herramientas informáticas que contribuyan y aporten a la evolución y desarrollo de nuestra sociedad. Es por ello la necesidad de desarrollar productos con calidad. La Ingeniería de Requisitos constituye un aspecto esencial en la obtención de un software con calidad. En ella se analizan, discuten y finalmente de definen los requisitos que la aplicación debe tener.

Bahamonde (1) define a la Ingeniería de Requisitos como la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema. Es de esta forma que una vez identificados de forma precisa y oportuna los requisitos por parte de los participantes en el proceso de negociación, se establece una fase de análisis para definir las prioridades de los mismos. En la fase de análisis de los requisitos, los stakeholders[1] tienen diferentes expectativas con respecto al sistema a desarrollar, ya sea por opiniones diferentes o por conflicto de intereses. Este proceso de toma de decisiones se manifiesta sobre la base de disímiles criterios: beneficio, costo, valor de implementación, riesgos, complejidad de los requisitos, dependencia entre los criterios, por citar algunos, que deben ser tenidos en cuenta para definir cuáles requisitos son necesarios implementar de forma prioritaria y cuáles pueden implementarse en una posterior versión del sistema de software.

Esta problemática ha sido abordada en diferentes trabajos, reportándose en la bibliografía diferentes técnicas y métodos que han sido empleados para establecer la prioridad de los requisitos de software durante el proceso de análisis de los mismos, desde el uso de métodos sencillos hasta la concepción de

[1] Participantes en el proceso de negociación

algoritmos heurísticos de gran complejidad. Algunas técnicas como: Proceso analítico jerárquico (AHP[2]), el Jerarquía AHP, la lista binaria de prioridad (BPL[3]), la Matriz de Wiegers, Juego de Planeamiento (PG[4]), la Técnica de los 100 dólares, así como muchos otros han sido empleados en este contexto. En este orden de ideas un gran número de trabajos se han enfocado en comparar estas técnicas para medir la eficacia de las mismas, identificando como parámetros comparativos, la facilidad de uso, escalabilidad, tiempo consumido, entre otros, todo con la finalidad de hacer la selección de una técnica en específico que satisfaga las necesidades de los stakeholder. De acuerdo con estos análisis comparativos (2), (3), (4), entre otros, se han podido segmentar en técnicas que poseen gran eficacia para bajos, medios y altos niveles de requisitos, pero ninguno define la técnica ideal para cualquier proceso de priorización. Cuando las expectativas del cliente son altas, los plazos son cortos, y los recursos son limitados, debe de asegurarse que el producto contiene las propiedades más esenciales (4). Establecer en cada paso la importancia relativa de los requisitos, permite una construcción secuencial del sistema para proporcionar un elevado valor del producto al menor costo posible.

Sin embargo no siempre se cuenta con las técnicas o herramientas para llevar a cabo el proceso de priorización de los requisitos o no siempre se emplea la técnica que realmente necesitan los stakeholders atendiendo a las características particulares del proceso de negociación. Existen una gran cantidad de técnicas, métodos que escoger entre ellas se convierte en un proceso complejo y lleno de incertidumbres. Es por eso que se plantea el **Problema a Resolver**: ¿Cómo disponer de un conjunto de técnicas, métodos y procesos para establecer prioridad de los requisitos de software, de forma tal que se viabilice el proceso de toma de decisiones durante la fase de análisis de los mismos y se tengan en cuenta los criterios y preferencias de los stakeholders?

Planteando así como **Objetivo General**:
Desarrollar una herramienta informática que soporte un conjunto de técnicas y métodos para establecer prioridad en requisitos de software.

[2] Por sus siglas en ingles AHP (Analytical Hierarchy Process).

[3] Por sus siglas en ingles BPL (Binary Priority Lists).

[4] Por sus siglas en ingles PG (Planning game) .

Desglosándolo en los **Objetivos Específicos**:

- Identificar el conjunto de atributos y/o criterios a tener en cuenta por los stakeholders para establecer prioridad entre los requisitos de software.
- Identificar cuáles son las técnicas, métodos y procesos más empleados para establecer prioridad entre los requisitos de software, de acuerdo al análisis bibliográfico.
- Realizar el análisis y diseño de la herramienta que apoye la toma de decisiones durante el proceso de análisis de los requisitos de software.
- Implementar la herramienta de apoyo a la toma de decisiones para la priorización de requisitos de software.
- Realizar las pruebas pertinentes para validar la herramienta.

Se plantea como **Idea a Defender**: Con el desarrollo de una herramienta informática que apoye la toma de decisiones durante el proceso de priorización de los requisitos del software es posible viabilizar el proceso de selección de los mismos teniendo en cuenta los criterios tanto de los clientes como de los desarrolladores del software y sus preferencias sobre el uso de las técnicas, métodos y procesos a emplear.

Para enmarcar los límites de la presente investigación se define como **Objeto de Investigación**: aplicación de métodos, técnicas y procesos en la fase análisis de requisitos de software. Delimitando el **Campo de Acción** en: aplicación de herramientas para la toma de decisiones a problemas de priorización y selección de requisitos de software en la UCI.
Trazando, para el cumplimento de los objetivos, las **Tareas de Investigación:**

Tareas de Tania Pérez Ramírez:

- Estudio del estado actual referente al empleo de las técnicas y métodos para establecer prioridad en el proceso de análisis de requisitos de software.
- Identificación de los atributos o criterios a tener en cuenta para el desarrollo del proceso de priorización y selección de los requisitos de software en la UCI.

- Selección de los métodos de priorización de requisitos incluyendo el modelo de programación lineal para el problema de priorización de requisitos, y un método de solución basado en algoritmos genéticos.
- Desarrollo del análisis y diseño de los métodos seleccionados incluyendo el modelo de programación lineal para el problema de priorización de requisitos, y un método de solución basado en algoritmos genéticos.
- Implementación de las técnicas, métodos y procesos descritos y de un algoritmo genético para la optimización del proceso priorización y selección de requisitos de software.
- Realización de pruebas de software a la herramienta desarrollada.

Tareas de Malena Perada Lemus:

- Estudio del estado actual referente al empleo de las técnicas, métodos y procesos para establecer prioridad en el proceso de análisis de requisitos de software.
- Diseño y Aplicación de una encuesta a los analistas, diseñadores y jefes de proyectos de la facultad 3 para obtener información referente a los atributos, técnicas, métodos y procesos que se emplean para priorizar requisitos.
- Identificación de los atributos o criterios a tener en cuenta para el desarrollo del proceso de priorización y selección de los requisitos de software en la UCI.
- Selección de los métodos de priorización de requisitos.
- Desarrollo del análisis y diseño de las técnica, métodos y procesos seleccionados.
- Implementación de los métodos descritos.
- Realización de pruebas de software a la herramienta desarrollada.

Entre los **métodos** a utilizar se encuentran:

- Análisis-síntesis de toda la información recopilada relacionada con esta temática.

- Histórico-Lógico para establecer una búsqueda sobre los estudios que se han desarrollado en la priorización y analizar y profundizar en cada uno de ellos.
- La Entrevista para investigar y entender de cómo se priorizan los requisitos de software en la UCI.

Proponiendo obtener los **Posibles Resultados**:

- Disponibilidad de una herramienta para la toma de decisiones que minimice las discrepancias entre los clientes y el grupo de desarrollo durante el proceso de priorización de requisitos de software.
- Refinamiento de los requisitos de software en función de las necesidades de las partes interesadas durante la fase de negociación y análisis del software.

La tesis se estructura en 4 capítulos. El primero aborda el marco teórico de la investigación. Se enfoca la priorización de requisitos de software como un proceso de toma de decisiones, se contextualiza el proceso dentro de la Ingeniería de Requisitos, se determinan los principales aspectos a tener en cuenta para priorizar, se describen las principales técnicas, métodos y procesos para la priorización de requisitos de software, además de establecer comparaciones entre ellos. En el capítulo 2 se describe la propuesta de solución donde se selecciona las técnicas, métodos y procesos a implementar a partir del análisis del capítulo 1 y de la descripción del proceso de priorización en la UCI, se definen las propiedades y características del sistema. En el capítulo 3 se documenta el proceso de implementación a partir de la creación de las Tareas de Ingeniería. Finalmente en el capítulo 4 se propone la validación del sistema a partir de la descripción de las pruebas de aceptación.

Capítulo 1. Fundamentación teórica

1. Introducción

En este capítulo se presenta un estudio sobre el proceso de priorización de los requisitos de software como un proceso de toma de decisiones. Se describe cómo se efectúa la priorización de los requisitos. Se clasifican las diferentes técnicas, métodos, procesos, utilizados en la temática, atendiendo los diferentes pasos que contemplan del proceso de priorización. Así como las diferentes herramientas de apoyo existentes. Por último se describe y enuncia las técnicas de programación a utilizar así como las herramientas con que se efectuará la aplicación.

1.1 La Priorización de Requisitos de Software como un Proceso de Toma de Decisiones. Conceptos

La Toma de Decisiones, es la capacidad de elegir un curso de acción entre varias alternativas. 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. La calidad de las decisiones tomadas marca la diferencia entre el éxito o el fracaso (5). El padre de la teoría de las decisiones Hebert Simón define en 1960 (6) que la Toma de Decisiones es un proceso de selección entre cursos alternativos de acción, basado en un conjunto de criterios, para alcanzar uno o más objetivos. El doctor Higuera (7) sintetiza al decir que tomar una decisión se refiere al proceso entero de elegir un curso de acción. Existen varios criterios, (6), (7), (8), para definir los pasos o fases. Todos concuerdan que para desarrollar el proceso de Toma de Decisiones primero se debe entender y definir el problema, definir los criterios de selección, es decir, los aspectos por los cuales los decisores se van a regir para tomar la decisión. Se deben identificar las alternativas, las opciones reales existentes para resolver el problema. Con los criterios se procede a evaluar las alternativas y escoger e implementar una alternativa, es decir se toma una decisión.

Azzolini (4) define al proceso de priorización de requisitos de software como proceso de decisión de naturaleza compleja realizado 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. Explica, con un previo análisis del SWEBOK (9), que el proceso de priorización implica identificar los actores del proceso de requisitos, sus

puntos de vista a partir de un análisis de la cultura y política organizacional para luego clasificar el conjunto de requisitos en función de su grado de prioridad, teniendo en consideración la posibilidad de implementar un esquema de negociación para resolver aquellos conflictos que puedan tenerse a lugar durante la ejecución del presente proceso. Resume la idea de varios autores de que la priorización de los requisitos de software ha sido reconocida como una de las actividades más importantes dentro del proceso de desarrollo del software además de considerarse que dicha actividad contribuye favorablemente en la toma de decisiones en los procesos de planificación, gestión y control del desarrollo de software. Agrega que no sólo el proceso de priorización de requisitos permitiría la disertación entre los que cumplan o no con los intereses de las partes sino que posibilitaría la planificación de futuras versiones con los paquetes de requisitos que queden. Constituyendo así la base para solucionar el Problema de la Próxima Versión (NRP[5]). Priorizar los requisitos de software implica conocer sobre el negocio, el problema que se desea resolver con la implementación de la aplicación, así como los actores o stakeholders del proceso de negociación donde se decidirá cuáles requisitos presentará la aplicación a partir de la clasificación de estos teniendo en cuenta la prioridad. Implica decidir cuáles requisitos deben y pueden ser escogidos y cuáles se evaluarían para las próximas versiones del producto. Los estudios de Berander, Khan, Lehtola (10) y Ngo, Ruhe (11) aseguran que la priorización de requisitos de software ha sido reconocida como una de las actividades de toma de decisiones más importantes en la Ingeniería de Requisitos. Es un proceso importante en la toma de decisiones para la obtener los requisitos esenciales de la aplicación que deberían ser priorizados con respectos a los demás. Galves (12) explica que los requisitos se distinguen unos de otros por el nivel de significatividad e importancia para los clientes y usuarios por lo que una correcta priorización entre ellos va a permitir desarrollar nuevas versiones del proyecto de forma continua sin verse retrasadas por tiempo en sus salidas.

Azzolini (4) desglosa el proceso de priorización en tres etapas:

- Selección de los criterios definidos para priorizar requisitos. Pueden ser criterios de negocios como necesidades de los usuarios, costo; o bien técnicos como factibilidad, recursos existentes, etc.
- Determinación de un ordenamiento de acuerdo a criterios específicos para uno o más participantes.
- Composición de un orden final combinando el punto anterior con varios participantes.

[5] Por sus siglas en inglés NRP (Next Release Problem)

De acuerdo con Azzolini (4) priorizar requisitos implica analizar el costo que implicaría la ejecución del proyecto teniendo en cuenta los recursos con los que el cliente dispone, la importancia que tiene el o los requisitos para el cliente, la complejidad técnica que presente el o los requisitos para los especialistas, entre otros criterios.

La priorización de requisitos constituye una parte importante en todo el proceso de desarrollo del software, a partir de ahí se define qué es lo que se implementará y cuáles son las características que la aplicación tendrá, no solo para la versión donde se encuentra también es la base para resolver el problema del NRP, favoreciendo la planificación y proyección de las próximas versiones. Por tanto, se puede decir que el proceso de priorización de requisitos constituye, sin lugar a dudas, un proceso de toma de decisiones. Para seleccionar el conjunto de requisitos que tendrá la aplicación se debe conocer las necesidades del cliente, el problema que se le resolverá, los criterios o aspectos por los que se evaluarán el conjunto de alternativas, los requisitos de software candidatos a ser escogidos.

1.2 Requisitos de Software

Requisitos de Software es la traducción en español de Software Requirements (13). El Glosario de Terminología Estándar de Ingeniería de Software (IEEE: Standard Glossary of Software Engineering Terminology) (14) define al requisito como:

a. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

b. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

c. Una representación en forma de documento de una condición o capacidad como las expresadas en a o en b.

La SWEBOK (Guide to the Software Engineering Body of Knowledge) en su versión del 2004 divide a la Ingeniería de Software en 10 áreas de conocimiento, ver figura 1, la primera es el área de conocimiento llamada Requisitos de Software y define a un requisito de software como una propiedad que tiene que ser expuesta para resolver un problema determinado del mundo real. Por lo tanto, es una propiedad que tiene que ser exhibida por un software desarrollado o adaptado para resolver algún problema en particular (9).

Se explica que uno de los atributos de los requisitos de software es la prioridad para permitir compensaciones frente a recursos limitados. En sí los requisitos de software son las características y funcionalidades que la aplicación debe tener. Se derivan de los requisitos del sistemas, aquellos que el cliente solicita.

Requisitos de Software
Diseño del Software
Construcción del Software
Prueba del Software
Mantenimiento del Software
Gestión de la Configuración del Software
Gestión de la Ingeniería del Software
Proceso de la Ingeniería del Software
Herramientas y Métodos de la Ingeniería del Software

Figura 1 Áreas del conocimiento de la SWEBOK (Fuente: SWEBOK tabla 1 página 1-2 (9)*)*

1.3 Requisitos Funcionales y no Funcionales

El sistema debe ser capaz de funcionar eficientemente y adaptarse a las condiciones donde será implantado; además debe cumplir una serie de requisitos relacionados con condiciones técnicas de trabajo, seguridad, calidad, etc. Los requisitos de software están divididos en funcionales y no funcionales. La SWEBOK (9) define a los requisitos funcionales como los que describen las funcionalidades que se ejecutarán del software. Indican características y restricciones sobre la funcionalidad del software. Definen el comportamiento interno del sistema. Constituyen las acciones que el sistema debe permitir realizar. Los no funcionales actúan sobre las restricciones de la solución y son conocidos como los requisitos de calidad.

Según Sommerville (15) los requisitos no funcionales son propiedades o cualidades que el producto debe tener y que lo hacen atractivo, usable, rápido y confiable. Los requisitos funcionales generalmente se relacionan con funciones específicas, mientras que los no funcionales suelen afectar a varias funciones. Según Berander (16), los no funcionales son propiedades que el sistema o las funciones del sistema deben tener por lo que los no funcionales son inútiles sin los funcionales. No obstante, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Sin embargo aclara que no siempre es viable priorizar los requisitos de ambas tipos juntos. Presentan características particularidades en concepto, funciones, en cómo se miden, que podrían influir a la hora de asignar prioridades entre dos requisitos de diferentes tipos. Aconseja, además en tener en cuenta este aspecto a la hora de escoger la técnica, método o proceso para priorizar.

1. 4 Contextualización de la Priorización de Requisitos de Software

La investigación realizada por Carlos Martín Azzolini (4) concluye que la priorización de los requisitos de software ha sido reconocida como una de las actividades más importantes dentro del proceso de desarrollo del software además de considerarse, que dicha actividad contribuye favorablemente la toma de decisiones en los procesos de planificación, gestión y control del desarrollo de Software. La SWEBOK explica que al área de conocimiento Requisitos de Software le concierne todo el proceso de elicitación desde elicitación, análisis, especificación, y finalmente validación de los requisitos independientemente de los otros tópicos que en dicha área se aborda como procesos de los requisitos. Sitúa la priorización en el análisis, en el que se analizan los requisitos extraídos con el objetivo de detectar, resolver conflictos entre ellos. Sommerville (15), Presman (17), Bahamonde (1), Toro (19) y el conjunto de autores del “CMMI for Development, Versión 1.2” (20) describen, en general, a la Ingeniería de Requisito como la disciplina encargada de la extracción, análisis y documentación de los requisitos. La dividen en diferentes fases aunque en resumen siguen el mismo procedimiento. Toro (19) y Bahamonde (1) contemplan en la fase de elicitación o extracción, la priorización de los requisitos pero es en la fase en la que se analizan, se estudia a profundidad los requisitos del sistema extraídos en la fase de elicitación; es donde se negocia con el cliente la gestión de dichos requisitos. Guiándose también por lEEE (22) en este trabajo se considera la actividad de priorización parte de la fase de análisis. A partir de la elicitación o extracción de los requisitos del sistema, se pasa a la fase de análisis y negociación, se analizan para a partir de ellos derivar los

requisitos de software. Es en ese momento en que se produce el proceso de priorización donde los stakeholders negocian y eligen los requisitos que tendrá la aplicación. Después se pasa a la fase de especificación de los requisitos elegidos y finalmente a la fase en la que se validan.

1.5 Aspectos a tener en cuenta en el proceso de priorización de requisitos

El proceso de priorización, requiere tener en cuenta aspectos que permitan la definición de los requisitos escogidos para la liberación del producto informático. Existen muchos criterios acerca de cuáles deberían ser esos aspectos, no todos se ajustan a todos los procesos de desarrollo del software. Para que un requisito sea atractivamente escogido depende de las necesidades de las partes interesadas, así como del ambiente de desarrollo del producto y principalmente depende de la forma en que los especialistas del sistema hayan decidido realizar la priorización. Sin embargo, existen aspectos que prevalecen en las investigaciones desarrolladas hasta la fecha por Azzolini (4), Berander (16) y Ma (23) y que las autoras de esta investigación consideran esenciales para el desarrollo del proceso:

Importancia: también llamado valor de negocio, es el aspecto por el cual los stakeholders pueden manifestar el grado de importancia que para ellos representa el requisito en cuestión. Sin embargo, el criterio “importancia” utilizado por un stakeholder en particular, puede identificarse por una serie de conceptos que no necesariamente resultan análogos a otro participante.

Penalidad: permitiría establecer por cada stakeholders lo que implicaría si el requisito en cuestión no fuera escogido.

Costo: el costo de un requisito puede ser presentado por el costo monetario y / o costos laborales. El costo laboral se expresa a través del de los especialistas, en horas o días, etc., que emplearían para desarrollar el requisito. Azzolini (4) explica que no solo permitiría contabilizar el costo monetario de los requisitos escogidos sino que reflejaría la complejidad y calidad.

Tiempo: se refiere al tiempo que implicaría implementar el requisito en cuestión.

Grado de Importancia de los stakeholders: se refiere a la influencia que tiene cada stakeholders en el proceso de negociación y priorización de los requisitos.

Riesgo: se valoran los requisitos atendiendo a riesgos a los que se podrían enfrentar los stakeholders. Berander (16) separa la volatilidad de los requisitos del riesgo. Sin embargo los cambios que podrían tener los requisitos a lo largo del proceso de desarrollo de software, constituye un riesgo, por lo que esta investigación lo considera implícito dentro del riesgo.

Dependencias: los requisitos se relacionan entre ellos. Es por ello que al escoger uno implica analizar con cuáles se relaciona en caso su elección afecte el valor, el costo de otro o que para ejecutarse necesite de otro u otros requisitos.

Consideran las autoras que estos son los aspectos principales en los que los stakeholders deben tener en cuenta en la priorización de los requisitos de software. Son criterios esenciales que influyen en la selección y disertación de cuáles son los requisitos "ideales" para implementar atendiendo a las características y condiciones en las que se desarrolla la aplicación y los criterios de los stakeholders.

1.6 Niveles del proceso de priorización de requisitos

Las técnicas, metodologías, métodos, desarrollados para priorizar los requisitos comprenden diferentes partes de este proceso. De acuerdo con Berander (10), Perini (24) y Vestola (2) la priorización se divide en niveles: actividades de priorización, técnicas, métodos y finalmente procesos. Los niveles altos utilizan los más bajos. Los autores basan su criterio de clasificación en los pasos del proceso de priorización que se enfocan las técnicas, metodologías, métodos, procesos que se han desarrollado.

1.6.1 Actividades

Se pueden realizar mediantes tres principales vías: asignando valores a los requisitos de acuerdo a criterios definidos, realizando la comparación por pares, o simplemente agrupándolos cada uno en una clase específica.

1.6.2 Técnicas

En este nivel se procesan los resultados de las actividades de priorización para calcular la prioridad de cada requisito basada en un solo aspecto, según Vestola (2). De acuerdo a cómo las presentan los resultados se dividen en tres escalas: nominal, ordinal y de ratio.

1.6.2.1 Escala Nominal

Las técnicas con escala nominal separan los requisitos en diferentes grupos según su la prioridad de cada uno. Los requisitos que se encuentran en un mismo grupo tienen la misma prioridad por lo que no brindan mucha información que permitan diferenciar cada requisito de los demás.

Asignación Numérica

Según Berander, Jönsson (25) y Berander, Svahnberg (26), la técnica de asignación numérica es la más común para la priorización de los requisitos. Es una técnica sencilla basada en la agrupación de los requisitos en grupos de prioridad diferentes. Creada por Brackett (1990) y recomendado posteriormente por el estándar 830-1998 IEEE (4).

Se basa en clasificar a los requisitos en tres categorías conceptuales como: Obligatorios, Deseables y No esenciales. De esta manera se indaga a los participantes respecto a la importancia percibida en cada uno de los requisitos especificados, utilizando la siguiente escala:

Categoría de Prioridad	Escala Numérica
No esencial	1
Deseable	2
Obligatorio	3

Tabla 1 Escala de Priorización de Requisitos (Fuente: Motupally (2008) (27)*y Ma (2009)* (23)*).*

Estudios empíricos demuestran que esta técnica no es tan efectiva cuando el número de requisitos es bajo (por ejemplo, 20 o menos) y que aproximadamente más del 80% de los participantes tienden a asignarlos, en la categoría “Obligatoria” y menos del 5% en la categoría “No esencial”, por lo cual, se pierde efectividad a la hora de decidir cuáles son los verdaderos requisitos prioritarios (25).

Existe una investigación realizada por Mikko Vestola (2) de estudios comparativos entre varias técnicas y métodos. En general, explica Vestola, esta técnica en comparación con las técnicas (AHP, entre otras) de comparación entre pares de requisitos, es menos exacta y más lenta para un promedio de 14 requisitos. Es decir, le es más complicado a los decisores dividir los requisitos en tres grupos según su valor de importancia. Sin embargo es la técnica más común (25) y (26). Se concluye que esta técnica es muy sencilla de emplear. Clasifican los requisitos ateniendo a la importancia que representan.

MoSCoW

Es una técnica para la priorización que se utiliza para decidir qué requisitos deben implementarse primero y cuáles después. Se utiliza esta técnica para dividirlos en cuatro categorías: M – MUST (deben estar), S – SHOULD (deben estar pero con menor prioridad que el anterior), C - COULD (pueden estar) y W - WON'T (no estarán). Hay definiciones muy específicas para cada una de estas cuatro categorías (28). El MoSCoW presenta en sí el mismo concepto que el de asignación numérica.

M – MUST : en esta categoría van a estar los requisitos de muy alta prioridad para el proyecto. Ellos deben ser parte de la solución final con el fin de que la solución se considere satisfactoria.
S – SHOULD: los requisitos de esta categoría también son de alta prioridad y son tan importantes como los requisitos del grupo anterior y estos se deben incluir en la solución si es posible.
C - COULD: estos requisitos son de menor prioridad. Ellos realmente no afectan a la solución de una manera u otra y se incluirán si el tiempo y los recursos lo permitan.
W - WON'T: estos requisitos pueden ser considerados para futuras versiones (futuras necesidades que las partes interesadas le gustaría tener), o se pueden excluir de la solución en conjunto.

En sí, es una variante de la asignación numérica salvo que separa en 4 grupos según la importancia de los requisitos. Esta técnica admite la priorización para los requisitos funcionales y no funcionales.

1.6.2.2 Escala Ordinal

El resultado es una lista ordenada. Estas técnicas permiten saber si un requisito es más importante que otro, pero en comparación con la escala ratio no es cuantitativa esa comparación.

Binary Priority List (BPL)

BPL, lista de prioridades binarias (en español) es una técnica utilizada para la priorización de requisitos, básicamente su descripción es igual al Árbol Binario de Búsqueda (BST). Es una técnica relativamente simple, de fácil utilización.

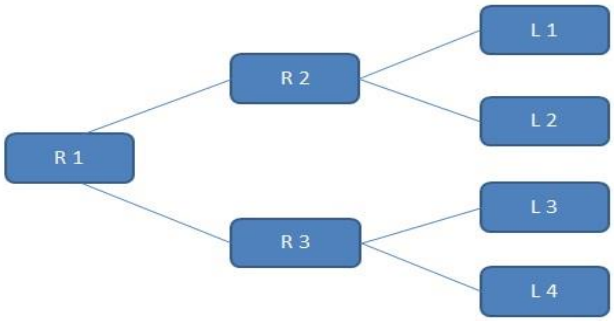

Figura 2 Estructura BPL (Fuente: (29)*)*

Según Bebensee, van de Weerd y Brinkkemper (29) la figura muestra una lista de los tres requisitos que a su vez tienen un número de sub-listas con más requisitos. De acuerdo con una estructura de árbol binario, los requisitos que se encuentran más arriba son más importantes que los que aparecen abajo. Por lo tanto, la prioridad R1 es menor que R2, pero los requisitos L2 y L4 tienen una prioridad más baja que el requisito de raíz, R2 y R3 respectivamente.

Las etapas de aplicación de esta técnica son:

1. Agregar a una pila todos los requisitos que se han recogido de diversas fuentes.

2. Tomar un elemento de la pila, que se utilizará como el requisito raíz.

3. Tomar otro de los requisitos y compararlo en términos de prioridad con el requisito raíz.

4. Si el requisito tiene una prioridad menor que el requisito raíz, se compara con el de debajo de la raíz y así sucesivamente. Si tiene una prioridad más alta que la raíz, se compara con el requisito

anterior de la raíz y así sucesivamente. Esto se hace hasta que el requisito finalmente se puede colocar como sub- requisito de un requisito.

5. Los pasos 2 a 4 se repiten para todos los requisitos.

6. Por último, recorrer la lista de arriba a abajo para conseguir el orden de prioridades de las necesidades de cada requisito.

Esta técnica se basa en la comparación entre pares de requisitos. En ningún estudio de los analizados, explican cómo calculan la prioridad. Quedaría por parte de los stakeholders definir el criterio de selección. No es recomendable emplearla para una gran cantidad de requisitos, Bebensee, Weerd y Brinkkemper (29) recomiendan utilizarla para una cantidad menor que 100. Comparan cada requisito con los demás, un grupo grande requisitos sería muy complicado su uso para los stakeholders. Vestola (2) aconseja que se utilice para una pequeña mediana y pequeña cantidad de requisitos pues es muy lenta por la comparación por pares.

Priority Groups

Fue introducida por Karlsson, Wohlin y Regnell en 1997 (30). Según Ma (23) el procedimiento de esta técnica es poner cada requisito en uno de los tres grupos: alta, media y baja prioridad. En caso de que un grupo tenga más de un requisito se deben crear tres nuevos subgrupos en donde estos son asignados. Este proceso se repite hasta que cada grupo tenga uno solo. Para una gran cantidad de requisitos la presentación de esta técnica sería complicada. Concordando con Qiao Ma (23) y Karlsson, Wohlin, Regnell (30) este técnica es muy difícil de usar y provee bajos resultados. Estos últimos la evaluaron para un pequeño grupo de requisitos. En comparación con las técnicas AHP, MST, Burbuja y BST es la de peor calificación en cuanto a facilidad de uso, fiabilidad y tolerancia a fallos. No existen estudios sobre su comportamiento para cantidad grande de requisitos de software. Ningún trabajo de los revisados explica cuál criterio de selección se emplea para asignar prioridades. Según los resultados arrojados por Vestola (2), Karsslon, Wohlin, Regnell (30) esta técnica presenta los peores resultados.

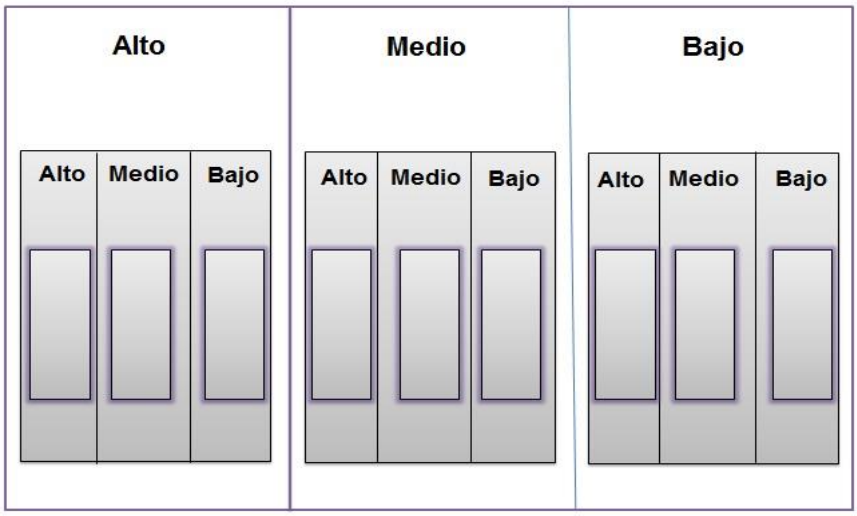

Figura 3 Priority Groups (Fuente: (2)*)*

Burbuja (Bubble Sort)

Consiste en ordenar los requisitos, tomándolos de dos en dos. Se solicita a los participantes que procuren una enésima ordenación de los requisitos y finaliza la ejecución del método, luego de obtener una lista ordenada según el valor de prioridad que se le asignó a cada uno (4). Los pasos necesarios para determinar la prioridad a las necesidades de la técnica burbuja son las siguientes (30):

1. Describir los requisitos en una columna vertical.

2. Comparar los dos principales requisitos de la columna entre sí, para determinar cuál es el más importante. Si el requisito inferior es más importante que el requisito superior entonces se intercambian de posiciones.

3. Repetir esta comparación por pares y el intercambio de los requisitos segundo y tercero, a continuación, tercero y cuarto; y así sucesivamente hasta que la parte inferior de la columna se alcanza.

4. Si alguno de los requisitos que se trasladaron durante las etapas de 2 y 3, se repite el proceso para toda la columna de partida de nuevo de los dos requisitos (paso 2).

El resultado del proceso es una columna ordenada de los requisitos donde el requisito más importante está en la parte superior de la columna y la menos importante es en la parte inferior.

La idea de la técnica burbuja es realizar la comparaciones por pares como el AHP. Ambas técnicas requieren n × (n - 1) / 2 comparaciones para n requisitos (30). Sin embargo, en la técnica burbuja, el que toma las decisiones sólo tiene que determinar cuál de los dos requisitos es más importante, no en qué medida como en AHP. Para la asignación de prioridades en ninguno de los escritos estudiado no existe ninguno que describa cómo determinar este valor.

Árbol de búsqueda binaria (BST)

BST, por sus siglas en inglés (Binary Search Tree). Se define como un algoritmo utilizado en la búsqueda de información y puede resultar fácil su adaptación para la priorización de requisitos (31). Todos los requisitos almacenados en el subárbol izquierdo del nodo x tienen menor prioridad que el requisito almacenado en el nodo x y este tiene menor prioridad que todos los almacenados en el subárbol derecho. El primer paso consiste en coger el primer requisito y convertirlo en el nodo raíz del árbol. Seguidamente se toma otro requisito y se compara con el nodo raíz. En caso de que tenga mayor prioridad se le agrega al nodo un hijo derecho, en caso contrario se agrega un hijo izquierdo y así sucesivamente se compararan todos los requisitos hasta que todos estén ubicados en el árbol. Todo este procedimiento se lleva un tiempo promedio de O (n log n). Finalmente se crea una lista ubicando los de menor prioridad al principio. Esta técnica está desarrollada bajo los mismos conceptos de comparación que el BPL. Se diferencian en la representación: el BPL se representa de forma horizontal, y el BST de forma vertical. Sin embargo la representación del BPL es más fácil de entender, los requisitos de mayor prioridad se colocan en el tope de la lista.

Planning Game (Juego de Planeamiento)

Constituye una técnica clave en la Metodología XP (Extreme Programming), (4) y (3). Clasifica los requisitos atendiendo a tres categorías: 1-“Sin ellos, el Sistema no funcionaría”, 2-“Son menos esenciales, pero al implementarlos se proveería de un significativo Valor de Negocio”, 3-“Sería deseable implementarlos”. Posteriormente se pasaría al establecimiento de un ranking en cada una de las categorías (4).

Es una combinación entre las técnicas Asignación Numérica y un Ranking Simple. Sin embargo el resultado final es una lista ordenada por el Ranking del grupo más importante de la Asignación Numérica por lo que las autoras consideran parte de la escala ordinal. Ahl (3) explica que se realizarán para la ejecución de este método n comparaciones, siendo n la cantidad de requisitos por lo que la complejidad computacional en tiempo de ejecución sería O(n) dependiendo de la cantidad n de requisitos. Sin embargo no contempla varios criterios de análisis, no tiene en cuenta la participación de varios expertos en la ejecución del método. Esta técnica, como el BPL, el BST, el Burbuja y el Priority Groups no se explica, en ninguna documentación estudiada, en qué se basan para determinar la prioridad de cada requisito. Para la metodología XP, que es donde se tiene mayor referencia que se utiliza, la prioridad la determina el cliente. Por tanto, teniendo en cuenta además la nomenclatura de los grupos, se concluye en esta investigación que la prioridad es determinada a partir del grado de importancia que para el cliente representa.

1.6.2.3 Escala Ratio

Estas técnicas dan la posibilidad de especificar las diferencias relativas de los requisitos, es decir cuantifican las diferencias entre ellos.

Técnica de los 100 puntos

Ahl (3) considera la probabilidad de que esta técnica, también conocida como Voto Acumulado (CV[6]), sea la más antigua y simple para la priorización, en otras bibliografías se refieren a dicha técnica como la de los 100 dólares o Voto Acumulado (4). Azzolini (4) y Carvajal y Hernández (32) coinciden en que esta técnica se basa en la asignación de puntos o dólares a los expertos dándoles la opción de votar por los requisitos que estos entiendan. Sin embargo los autores discrepan en la cantidad de puntos o dólares a repartir: Azzolini deja la cantidad indefinida, Carvajal y Hernández (32) la restringen a 100 puntos. La técnica de los 100 puntos consiste en sí en asignar una suma de puntos o dólares a cada stakeholder para que cada uno le asigne a cada requisito, según entienda, parte de los puntos o dólares que le han sido asignados. Posteriormente se pasa a la tabulación de esas asignaciones individuales en un diagrama de frecuencias relativas obteniendo como resultado final el porcentaje de importancia o prioridad de cada

[6] Por sus siglas en inglés CV (Cumulative Voting)

requisito con respecto a los demás. Esta técnica tiene en cuenta el grado de importancia de los stakeholders al asignarle una x cantidad de dinero o puntos según el experto que sea. De forma general, se basa en establecer escalas para clasificar los requisitos. Es muy fácil de usar, aunque no sería de mucha utilidad a la hora de analizar una gran cantidad de requisitos. Sin embargo, podría ser sensible a las llamadas "tácticas astutas", lo que significa que las partes interesadas distribuyen sus puntos basado en cómo piensan que otros lo harán con el fin de obtener una alta prioridad a sus requisitos preferidos.

Votación jerárquica acumulativa (HCV[7])

VHC fue desarrollado para responder a los problemas de escalabilidad de la técnica CV o método de los 100 puntos. La idea de HCV es básicamente la misma que CV. En otras palabras, el establecimiento de prioridades con el HCV se realiza distribuyendo los puntos a los requisitos. Sin embargo, cuando se utiliza el HCV, no todos los requisitos son priorizados al mismo tiempo. Se clasifican a diferentes niveles de jerarquías y la priorización se realiza sólo dentro de cada nivel de jerarquía. La figura 4 muestra un ejemplo de cómo utilizar el HCV.

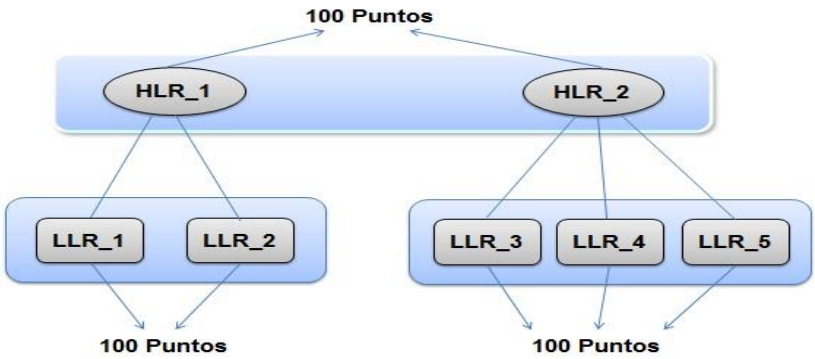

Figura 4 Procedimiento del HCV según Berander y Jönsson (Fuente: (25)*)*

Este ejemplo contiene dos niveles de requisitos: requisitos de alto nivel (HLR) y los requisitos de bajo nivel (LLR). Los requisitos dentro de un bloque de prioridades (las zonas grises) son priorizados juntos. En lugar de priorizar los cinco requisitos de bajo nivel al mismo tiempo como en la CV, los requisitos se

[7] Por sus siglas en inglés HCV (Hierarchical cumulative voting)

dividen en dos bloques de prioridades (2). En comparación con el CV o 100 puntos, es recomendable para una mayor cantidad a dividir en diferentes grupos y comparar en cada uno de ellos.

AHP (Analytic Hierarchy Process)

El Proceso de Análisis Jerárquico (PAJ) , por sus siglas en español, fue creado por Thomas L. Saaty en 1980 (33) e introducido por Karlsson en 1996 (4) como técnica en la priorización de requisitos. Weiss (34) define el AHP como una técnica de análisis de decisión usada para evaluar alternativas atendiendo a varios atributos por uno o más actores que presentan objetivos a cumplir basándose en la descomposición jerárquica del problema en diferentes subproblemas con el fin de facilitar su comprensión y evaluación. Wiegers (35) define al AHP como una teoría de medición, que enfatiza el uso de estructuras elaboradas para representar un problema de decisión emulando lo que parece ser un método innato de operación de la mente humana; permite una repetición de este proceso del cerebro, agrupando los elementos en lo que se consideran grupos que comparten propiedades comunes, generándose un nivel dentro del sistema. Esos elementos pueden, a su vez, agruparse de acuerdo a otro tipo de propiedades, generando los elementos de otro nivel “más alto” con un único elemento el cual es identificado frecuentemente como el objetivo o meta del proceso de decisión que se está llevando a cabo. Otros autores Doorn, Hadad y Kaplan (36) describen al PAJ como una técnica que involucra una jerarquía de criterios de decisión claves y luego hace comparaciones entre cada posible par de cada grupo. Su aplicación específica en la priorización de requisitos se basa en los elementos costo y valor relativos (37). Para todos los pares de requisitos candidatos se estima un valor y un costo de implementación que permite la comparación de ambos requisitos. Basados en esta información, los jefes de proyecto pueden tomar decisiones como qué requisitos de software pueden ser postergados o excluidos para mantener el tiempo de entrega del producto a un mínimo.

Descripción del método

Los analistas y diseñadores del sistema de software expresan su criterio con respecto a cada requisito en función de los demás, este proceso se hace comparando cada par de requisito, en función de los criterios definidos por los stakeholders para establecer la prioridad, el cual consideran mediante una escala definida, ver tabla 2. Son representados mediante una matriz, donde los requisitos se encuentran en las

fila i y en la columna j. Ya sea Si, Sj, Wi o WJ; n o k la cantidad de requisitos, tienen que cumplir las propiedades:

Sii = 1; para todo i=1,k

Sij = 1/Sji; para todo i=1,k

Una vez que le matriz tenga todos los valores se pasa a la normalización de cada celda mediante la división de cada uno de los valores de la matriz por la respectiva suma de la columna en donde la celda se halle. Seguidamente se pasa a la estimación del vector de valor ponderándolo por el número de requisitos a priorizar, proporcionando el vector de preferencia del stakeholder. O sea se multiplica el vector resultante del paso anterior por el recíproco de la cantidad de los requisitos a priorizar (n) obteniendo así los porcientos de importancia o prioridad de cada requisito. El valor de cohesión o el radio de consistencia (CR) entre los stakeholders participantes se calcula: CR=CI/RI. CI (índice de consistencia) se calcula: CI=(λmax-n)/(n-1) donde λmax es el máximo valor que adquiere la matriz de comparación y se calcula: primero multiplicando el vector de los valores de prioridad o importancia de los requisitos por la matriz original, segundo dividiendo el vector resultante del paso anterior por el vector de porcientos de importancia o prioridad y finalmente el tercer paso para calcular λmax es la división del resultado de la suma de cada valor del vector resultante del paso anterior entre n, la cantidad de requisitos a priorizar. El RI[8] (índice aleatorio) adquiere un valor estocástico que se encuentra especificado en función de la cantidad de opciones existentes que serán analizadas. En Saaty (1980), existe una tabla que tabula los valores de RI para un conjunto de 25 opciones (4). Saaty (33) y (37) concuerdan con que el índice (CR) sea menor o igual que 0.10, la inconsistencia exhibida será aceptable. Aunque, Karlsson y Ryan aclaran que en la práctica frecuentemente es mayor de 0.10. Según el estudio realizado por Ahl (3) el AHP no soporta la comparación de muchos requisitos toma n(n-1)/2 comparaciones para n requisitos constituyendo $O(n^2)$ la complejidad computacional del algoritmo del AHP. Comparados con los anteriores es lento en tiempo de ejecución hablando. No obstante el PAJ permite hacer la comparación de los requisitos por varios expertos. Sin embargo Jung (38) asegura que esta técnica puede ser inefectiva cuando se enfrente a sistemas con muchos requisitos, o exista mucha paridad en criterios en la priorización. Esta técnica proporciona los medios para comprobar la exactitud de las comparaciones

[8] Por sus siglas en inglés RI(indices of randomly)

mediante el cálculo de la relación de coherencia. Brinda mayor información, calcula el valor relativo de los requisitos. Permite una granularidad del valor de importancia.

Intensidad de la Importancia	*Definición*	Explicación
1	Importancia Igual.	Dos requerimientos contribuyen por igual al objetivo de satisfacción
3	Importancia Moderada.	La experiencia y el juicio favorecen levemente un requerimiento por sobre otro
5	Importancia Esencial o Fuerte.	La experiencia y el juicio favorecen fuertemente un requerimiento por sobre otro
7	Importancia Demostrada.	Un requerimiento se favorece fuertemente y su dominación es demostrada en la práctica.
9	Importancia Extrema.	La evidencia que favorece a un requerimiento por sobre otro está en el orden más alto posible de la afirmación
2,4,6,8	Importancia contenida en dos juicios adyacentes	Valores intermedio entre dos juicios adyacentes.

Tabla 2 Escala utilizada para la valorización de Requisitos de Software (Fuente: (39)*).*

Jerarquía AHP

Fue introducida por Karlsson (2). La idea básica de esta técnica, es estructurar los requisitos en una jerarquía donde se contemple el grado de generalización de los requisitos. Es decir, requisitos más generales, se les asignará un nivel más alto en la Jerarquía que aquellos requisitos más específicos. Luego de establecer dicha Jerarquía, se aplicará el método AHP para realizar comparaciones entre pares en cada uno de los niveles jerárquicos establecidos. Esta técnica reduce las comparaciones entre pares en forma significativa, ya que no se efectuaría sobre la totalidad de los requisitos disponibles como en AHP (puro), sino que, se establecerían sobre las segmentaciones que se establecen en cada uno de los niveles de la jerarquía. Donde su desventaja principal estaría dada por la dificultad que implicaría detectar la inconsistencia en los juicios personales de los participantes, según (4). Sin embargo, tiene desventaja en la identificación de errores al reducir el número de comparaciones redundantes.

MST (Minimal Spanning Tree)

El origen del MST se remonta a 1926 por Otakar Boruvka para buscar la solución más económica en la construcción de una red eléctrica por la Electrical Power Company of Western Moravia in Brno. Es introducido a la priorización de requisitos en 1998 (30). Consiste en evitar la redundancia, disminuyendo así el número de comparaciones: n-1 para n requisitos. Karlsson, Wohlin y Regnell (30) fundamentan que esta técnica elimina la redundancia de las comparaciones entre los requisitos que se hacen empleando la técnica del AHP. Describen en tres etapas al MST:

1. Como preparación, la construcción de un árbol de expansión mínima para el esquema n-1 pares únicos de requisitos.

2. En la ejecución, se realiza la comparación de estos pares de requisitos.

3. Se calcula las intensidades de falta de importancia tomando la media geométrica de las existentes intensidades de todas las formas posibles en las que están conectados. Y después se aplica el AHP.

El árbol de expansión mínima requiere poco esfuerzo además de poco tiempo de ejecución sin embargo Ma (23) asegura que no es hábil para identificar inconsistencias. Aunque es válido aclarar que este método sería útil usarlo en caso de que los expertos tomaran decisiones compatibles, que generalmente no es el caso. Por otra parte, es más sensible a los errores de juicio ya que toda redundancia ha sido eliminada.

Planning Game combinado con AHP

Constituye una combinación entre técnicas de diferentes escalas. Sin embargo, el resultado final es la del AHP. La idea de combinar estas técnicas es coger los beneficios de cada una. La posibilidad que da el AHP de calcular la inconsistencia de criterios y entre expertos, además de comparar por pares los requisitos con la rapidez y facilidad del PG. La propuesta de Karlsson, Berander, Regnell y Wohlin (40) es dividir los requisitos en tres pilas según las categorías del PG. Después se escoge la pila más importante y se le aplica el AHP para priorizar los requisitos. Concordando con Ahl (3), este híbrido le posibilita al especialista dirigir su mayor atención a los requisitos más importantes y que el mejor número de

comparaciones sería n para n requisitos. Ahl en su tesis de maestría (3) recomienda, basándose en análisis matemáticos, usar esta combinación de técnicas para cuando se tengan más de 11 o 21 requisitos suponiendo que para los stakeholders el 80 o 90 por ciento de los requisitos son importantes aunque no siempre se puede juzgar desde el principio.

1.6.3 Métodos

Son más sofisticados que las técnicas, consideran más de un aspecto. Emplean las técnicas para calcular la prioridad.

Wiegers Matrix Approach (Matriz de Wiegers)

Este método calcula las puntuaciones de prioridad relativa para obtener una lista de los requisitos mediante la estimación de cuatro criterios de priorización: costo, riesgo, penalidad relativa y valor relativo; contiene el mayor número absoluto de conceptos comparables. Ahl (41) y Wiegers (35) lo describen en 8 pasos :

Paso 1**:** listar todos los requisitos, características, o casos de uso que desea dar prioridad en una hoja de cálculo.

Paso 2: estimar el beneficio relativo que cada característica proporciona al cliente o a la empresa en una escala de 1 a 9, donde 1 indica muy pocos beneficios y 9 siendo el máximo beneficio posible. Estos beneficios indican la alineación con los requisitos del negocio del producto. Los representantes de los clientes son las personas más indicadas para juzgar estos beneficios.

Paso 3: estimar la penalidad en relación al negocio del cliente que sufrirían si la función no está incluida. Una vez más, utilizar una escala de 1 a 9, donde 1 significa esencialmente ninguna penalización y 9 indica un inconveniente muy grave.

Paso 4: la columna Valor total es la suma del beneficio relativo y la penalidad.

Paso 5: estimar el costo relativo de la ejecución de cada función, una vez más en una escala que va desde un mínimo de 1 y un máximo de 9. La hoja de cálculo calculará el porcentaje del coste total de cada

función. Los desarrolladores estiman las clasificaciones de costo basado en factores tales como la complejidad del requisito, en la medida de trabajo de la interfaz de usuario requerida, la capacidad potencial de reutilización diseños o código existentes, y los niveles de pruebas y documentación necesaria.

Paso 6: los desarrolladores estiman el grado relativo del riesgo técnico o de otro tipo asociados a cada función en una escala del 1 al 9. Una estimación de 1 significa que puede programar en su sueño, mientras que el 9 indica serias preocupaciones sobre la viabilidad, la disponibilidad de personal con la experiencia necesaria, o el uso de herramientas y tecnologías no probadas o no familiares. La hoja de cálculo calcula el porcentaje del riesgo total que proviene de cada característica.

Paso 7: una vez que se entra en las estimaciones en la hoja de cálculo, se calcula un número de prioridad para cada requisito. La fórmula para la columna de prioridad es: prioridad = valor% / (% del costo * Peso costo + riesgo% * ponderación de riesgo).

Paso 8: Ordenar la lista de características en orden descendente de prioridad calculado. Las características de la parte superior de la lista tienen el equilibrio más favorable de valor, costo y riesgo, y por lo tanto deben tener mayor prioridad. La clave del éxito consiste en que los stakeholders deben revisar la hoja de cálculo completa para llegar a un acuerdo sobre las calificaciones y la secuencia resultante.

VOP[9] **(Priorización Orientado al Valor)**

Azar, Smith y Cordes (42) describen este marco de trabajo desarrollado bajo la forma de la matriz de Weigers. VOP ofrece toda la visibilidad de las partes interesadas en la toma de decisiones. Se divide en dos pasos:

1. Establecimiento del marco de trabajo.

[9] Por sus siglas en inglés VOP (Value-Oriented Prioritization)

2. El negocio, entonces utiliza esos valores fundamentales para ayudar en la priorización de necesidades o la aplicación del marco.

El marco establecido en el primer paso se utiliza para identificar los valores fundamentales de la empresa. Los valores fundamentales se establecen en el nivel ejecutivo de la organización, y son independientes de los proyectos individuales, los requisitos individuales y grupos de usuarios individuales. Ejemplos de valores fundamentales incluyen el valor de:

I. ser líder de mercado en términos de ventas;

II. ser de ser el primero en el mercado con un producto;

III. retención del cliente;

IV. atraer nuevos clientes;

V. incorporando la última tecnología.

Señalan los autores del VOP (42) que estos son posibles valores que una organización puede expresar. Posteriormente los ejecutivos dela organización deben proporcionar una idea de la importancia de esos valores a la organización mediante la asignación de pesos que utilizan una simple escala ordinal de 0 (no importante) a 10 (Crítico). Se identifican y ponderan los riesgos. Se ponderan utilizando una escala negativa.

Azar, Smith y Cordes (42), proponen la matriz para la ponderación de los valores (Tabla 3) que incorpora cinco valores comerciales y dos riesgos técnicos, $V_{0,i}$ es el peso del valor de negocio i. $R_{0,}$j es el peso de riesgo j. $W_{i,j}$ es el peso asignado a r_i requisito con respecto al valor del negocio V_j. $W'_{i,j}$ es el peso asignado a la r_i requisito con respecto R_j.

	Valores de Negocio (V1……Vn)					**Riesgos(R1….Rm)**		
Rq	**Ventas**	**Marketing**	**Competitividad**	**Estrategia**	**Retención**	**Técnicos**	**Negocios**	**Punt**

mt					del cliente			uación
	V_1=7	V_2=6	Vi=8	Vi+1=10	Vn=7	R1=-8	Rm=-5	
r_1								
r_2			$W_{i,j}$			$W'_{i,j}$		
…								
r_n								

Tabla 3 Matriz de Priorización del Valor Orientado (Fuente: (42)*)*

Finalmente se pasa a calcular la puntuación de cada requisito (42):

$$\forall r \in \{R\}: Sr = \sum_{i=1}^{n} (Vi \cdot Wr, i) + \sum_{j=1}^{m} (Rj \cdot W'r, j)$$

Enfoque Costo-Valor para la priorización de requisitos

Desarrollado por Karlsson y Ryan (37) en 1997. Utilizan la técnica AHP para calcular el valor y costo de implementación relativo de cada requisito. Posteriormente se grafican el par (costo, valor). La estrategia de seleccionar requisitos prioritarios, consta en establecer aquel conjunto de requisitos cuya prioridad relativa sea mayor, sujeta a la restricción del costo total de su respectiva implementación.

El problema se resuelve mediante los tres pasos que se muestran a continuación:

1. Utilizar el AHP se asigna el valor y costo relativo a cada requisito para el desarrollo del sistema de software.

2. Trazar la relación costo-valor para cada requisito en un plano xy. Los ejes X y Y denotan la relación entre el costo y el valor relativo, respectivamente.

3. Se inspeccionan los puntos de la función de valor-costo para decidir cuáles de los requisito tendrá la aplicación.

Si un requisito tiene un costo relativamente alto y un bajo valor relativo, el mismo quedará descartado a la hora de implementar el sistema de software.

El paso 3 depende de la inspección intuitiva de los puntos de la función costo-valor en el plano xy. Sin embargo esa comparación no garantiza que se escojan las mejores soluciones o las más aceptables para ello se hace necesario plantearlo como un problema de optimización multiobjetivo que permita satisfacer la necesidad de lograr una mayor cantidad de requisitos con valores de importancias altos a un mínimo costo. Este método resulta conveniente emplearlo para una pequeña o mediana cantidad de requisitos. Decidir en una gráfica un conjunto de soluciones sobre un gran número de puntos resulta engorroso.

1.6.4 Procesos

Tienen en cuenta a la descripción de las etapas que se necesitan en la organización para llevar a cabo el proceso de priorización. Incluye factores como las diferencias entre los stakeholders, cómo afecta la selección de los requisitos en el desarrollo del software.

Estrategia Cognitiva

Se basa en los conocimientos que tengan los participantes del proceso de priorización de requisitos. Según Carod (43) se definen tres variables para que un participante asigne un valor a un requisito:

-*Conocimiento del individuo sobre el requisito***:** se asignarán 4 niveles donde cada nivel tendrá un peso determinado (sin conocimiento, poco conocimiento, conocimiento suficiente y experto).

-*Categorización del individuo*: considera la jerarquía del individuo dentro y fuera de la organización. Como en toda organización existen múltiples niveles de jerarquías, una asignación válida de peso estará dada por un peso diferente para cada nivel. Para el caso de los participantes que están fuera de la organización (desarrolladores, clientes) la asignación será un valor específico entre los valores considerados en la organización.

-*Valor asignado*: valor entero que pertenece al conjunto {-9,-8,…-1, 0, 1,…,9}. Cuando el valor es negativo significa que la implementación de dicho requisito influye de manera negativa en la implementación del sistema.

En este ejemplo se observan los valores finales de dos requisitos (req1 y req2):

Requisito	Participante	Valor Prioridad	Categoría	Conocimiento	(PC) Peso	(VF) Valor

					Cognitivo	Final
Req 1	P1	6	2	1	9	
	P2	7	1	3	11	
	P3	8	2	3	13	39/4=9.75
	P4	-6	3	9	6	
Req 2	P2	-1	1	9	9	
	P3	7	2	3	12	21/2=10.5

Tabla 4 Estrategia Cognitiva (Fuente: Nadina Martínez Carod (43)*)*

Proceso de Optimización Multiobjetivos

Un problema de optimización multi-objetivo se puede definir como el problema de encontrar un vector de variables de decisión que satisfaga ciertas restricciones y optimice un vector de funciones cuyos elementos representan las funciones objetivo (44). El término optimización se refiere a la búsqueda de una o varias soluciones tal que contenga valores aceptables para todas las funciones objetivo (45). Existen problemas clásicos multiobjetivos, o monobjetivos adaptados, a los que el proceso de la priorización se pudiera ajustar. Los estudios que tratan al proceso de priorización como un problema de optimización (46), (47), (48), (49) se basan en el problema clásico de la mochila. El problema de la mochila se basa en el beneficio o valor de cada artículo de la mochila con restricciones de peso, ajustándose al proceso de priorización los requisitos atendiendo al valor de ejecución que presentan y al costo que implicaría su desarrollo.

Descripción del problema de la mochila para la optimización de los requisitos de un software.

Analizado el trabajo para la priorización de requisitos Jung (38) describe el problema multiobjetivo :

Para j=1 hasta n, la variable de decisión es definida por:

$$x_j = \begin{cases} 1 \text{ si el requisito j es incluido en la mochila.} \\ \end{cases}$$

0 si no está incluido.

El problema de la mochila para el paso 1 (P1) puede ser representado de la siguiente forma:

P1

$$\text{Max } Z = \sum_{j=1}^{n} cjxj$$

Sujeto a:

$$\sum_{j=1}^{n} cjxj \leq b$$

$$xj = 1 \text{ o } xj = 0, \text{para } j = 1, \ldots, n.$$

En P1 tanto $x_j *$ como $Z_0 * = \sum_{j=1}^{n} v_j x_j$ representan la solución óptima y el valor óptimo

El problema 0/1 de la mochila para el paso 2 puede ser representado de la siguiente manera:

P2

$$\text{Min } Y_0 = \sum_{j=1}^{n} cjxj$$

Sujeto a:

$$\sum_{j=1}^{n} v_j x_j \leq Z_0 *$$

$$xj = 1 \text{ o } xj = 0, \text{para } j = 1, \ldots, n.$$

En P2 tanto $x_j *$ como $Y_0 * = \sum_{j=1}^{n} cjxj$ representan la solución óptima y el mínimo costo relativo.

En resumen se describe el problema de la mochila en:

Paso 1: Hallando el máximo valor relativo.

Maximizar la suma de los valores relativos de los requisitos a implementar de tantos requisitos sea posible teniendo en cuenta las limitaciones de costos. Esto es modelado como el problema P1 del problema de la mochila. Resolviendo este problema de la mochila con un determinado costo b se genera el valor óptimo (máximo valor relativo) $Z_0 *$. Este valor es utilizado en el Paso 2.

Paso 2: Hallando el mínimo costo relativo.

Para la solución óptima x_j* de P1 es el mismo costo óptimo (Y_0*) en P2 entonces Z_0* y Y_0* se convierten en el máximo valor relativo y el mínimo costo relativo de los requisitos respectivamente.

Li, van den Akker y Brinkkemper (46), además de Greer y Ruhe (48) se enfocan para una sola función objetivo. Zhang, Harman, Finkelstein, y Mansouri (49) modelan el problema atendiendo a la opinión de varios stakeholders en vista de maximizar el valor minimizando el costo. Existe un trabajo más reciente "Empirical evaluation of search based requirements interaction management" (47) que amplía la formulación del problema donde recoge diferentes factores. Plantea el problema de la mochila queriendo maximizar la cantidad de requisitos atendiendo a sus valores de importancias minimizando el costo y teniendo en cuenta el grado de importancia de los stakeholders, así como dependencias entres los requisitos.

Zhang, Harman y Lim (47) detectan 6 principales dependencias entre los requisitos:

- *and*: para que exista debe cumplirse que el requisito R1 es seleccionado el R2 también.

- *or*: si el requisito R1 es seleccionado debe cumplirse que el R2 no, o viceversa.

- *precedencia:* el requisito R1 debe ser seleccionado y desarrollado primero que el R2.

- *valor relativo*: si el requisito R1 es seleccionado entonces afecta el valor de R2 por el stakeholder.

- *costo relativo:* si el requisito R1 es seleccionado entonces afecta el costo de R2 por el stakeholder.

Formulación del problema

Se denota los requisitos por $R = \{r_1 \dots., r_n\}$ donde n constituye la cantidad de requisitos. Al mismo tiempo se denota los stakeholders por $C = \{c1 \dots., cm\}$ donde m constituye la cantidad de stakeholders. Cada uno de los stakeholders tiene un grado importancia relacionado denotado por $W = \{w1 \dots., wm\}$ donde $\sum_{j=1}^{m} w_j = 1$.

El costo se denota por $\text{Cost} = \{\text{cost}_1 \ldots, \text{cost}_n\}$ de cada requisito n. Cada valor que asigne el stakeholder (c_j) para cada requisito (r_i) está dado por el par $v(r_i, c_j)$ donde es mayor que 0 si el stakeholder decide que se implemente, 0 en caso contrario. Es así como se define la puntuación (p) de cada requisito:

$$p_i = \sum_{j=1}^{m} w_j \cdot v(r_i, c_j)$$

Como resultado se plantea el problema multiobjetivo:

$$\text{Maximizar } f_1(\vec{x}) = \sum_{i=1}^{n} p \cdot x_i$$

$$\text{Maximizar } f_2(\vec{x}) = -\sum_{i=1}^{n} \text{cost}_i \cdot x_i$$

El vector de decisión determina los requisitos que serán seleccionados, toman valor 1 en caso que sea escogido y 0 en caso contrario $x = \{X_1, \ldots, X_i, \ldots. X_n\} \in \{0,1\}$

El problema está sujeto a las restricciones derivadas de las dependencias descritas

-para la restricción del and: $x_i = (i,j) \in \xi$

- para la restricción del or: $x_i = x_j \vee x_i = xj = 0 = (i,j) \in \varphi$

-para la restricción de la precedencia: $x_i >= x_j(i,j) \in \chi$

Las dependencias de costo y valor relativo (47) no se pueden describir como restricciones por lo que se hacen las modificaciones en las funciones objetivo:

Si se cumple la dependencia del valor relativo:

$xi = xj = 1\, r(i,j) \in \psi \Rightarrow$

$$f1(\vec{x}) = \sum_{k=1}^{n} pk \cdot xk + (pi + pj) \cdot IF$$

Si se cumple la dependencia del costo relativo

$xi = xj = 1\, r(i,j) \in \omega \Rightarrow$

$$f_2(\vec{x}) = -\sum_{k=1}^{n} cost_k \cdot x_k + (cost_i + cost_j) \cdot IF$$

IF es el factor de influencia determinado por los valores $\{-0.4, -0.2, 0.2, 0.4\}$

Estos valores de dependencias entre los requisitos se obtendrán de la comparación por pares que brinda el AHP como mismo se obtendrán los valores de grado de importancia de los stakeholders, costo y valor de importancia modificando así el trabajo realizado por Karlsson y Ryan (37).

Propuesta de solución del problema

El problema planteado se resuelve mediante la modificación del algoritmo genético NSGA II[10], algoritmo genético de clasificación no-dominado. El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin. Colombo y Mumford (50) explican que estos algoritmos basan su funcionamiento principalmente en simular computacionalmente los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno.

Se utilizan alternativamente los términos cromosoma y genotipo para describir a un conjunto de parámetros que codifican una posible solución al problema. El término gen hace referencia a una particular entidad funcional de la solución, por ejemplo, a un parámetro específico en un problema de optimización. La adaptación del individuo se corresponde con la calidad de la posible solución mientras que el entorno, se corresponde con el problema a resolver.

A partir de una población inicial se seleccionan los individuos que formarán parte de la nueva población que sufrirá la fase de cruzamiento y finalmente mutará. Se han recogido estudios (50), (52), (53), (54) que emplean los algoritmos genéticos como métodos de solución al problema de la mochila. Son métodos populares para la generación de soluciones óptimas de Pareto a un problema de optimización multiobjetivo. El NSGA II es un popular algoritmo genético basado en la no-dominación para la optimización de objetivo múltiple o multicriterio. La población se inicia como de costumbre. Una vez que la

[10] Por sus siglas en inglés NSGA-II (Non-dominated Sorting Genetic Algorithm-II)

población ha inicializado la población está ordenada basada en la no-dominación en cada frente. El primer frente es completamente set no dominante en la población actual y el segundo frente está dominado por los individuos sólo en la primera parte delantera y la parte delantera va así sucesivamente. Cada individuo en el cada frente son valores de clasificación (fitness) asignado o en base a frente en las que pertenecen. Las personas en primer frente se les da un valor de aptitud de 1 y de los individuos en la segunda se les asigna el valor de fitness, 2 y así sucesivamente. La priorización de requisitos de software es un proceso de toma de decisiones. Ya sea de forma empírica o basándose en conocimientos teóricos, los stakeholders analizan las necesidades del cliente, el problema que se desea solucionar, definen los criterios de selección y finalmente toman la decisión de desarrollar un paquete de requisitos previamente analizados y priorizados en correspondencia con los aspectos o criterios establecidos. El proceso de priorización se dividen en niveles, según los niveles o pasos dentro de dicho proceso que son considerados. Es decir, las actividades están enfocadas solamente en la ponderación de valores a partir de un criterio definido o simplemente clasifican los requisitos, las técnicas además de ponderar, calculan estos valores y dan un resultado. Los métodos permiten calcular esos requisitos atendiendo a más de un criterio predefinido y los procesos no solo tienen en cuenta varios criterios, permiten analizar las diferentes opiniones de los stakeholders y así como el grado de importancia de estos o la influencia de ellos en la selección de los requisitos. Las circunstancias en que se desarrolla un producto informático son particulares, así como las perspectivas y conocimientos de los desarrolladores. El empleo de alguna técnica, método o proceso depende de estos factores.

1.7 Herramientas existentes

En múltiples ocasiones ha sido necesario tomar decisiones, por lo que es un proceso muy estudiado. Con el objetivo de facilitarlo han surgido herramientas que apoyen a los decisores brindándole información necesaria. Una herramienta de apoyo a la toma de decisiones según Finlay (55) está basado en computador que ayuda en el proceso de toma de decisiones. Con este fin, especialistas del Instituto Superior Politécnico José Antonio Echeverría (CUJAE) en el 2004 desarrollaron una herramienta llamada Jerarquías Versión 1.0. Se basa en el AHP clásico pero con varias modificaciones: soporta un sistema multi-experto para ello calculan el promedio de todos los valores de todas las matrices, multicriterio, además de permitir realizar comparaciones entre los criterios y así definir el valor de importancia de estos. Muchas de las aplicaciones del AHP nunca se reportan al público en general, porque tienen lugar en las

altas esferas de las organizaciones donde la seguridad y las consideraciones de privacidad prohíben su divulgación. Sin embargo, algunos usos se discuten en la literatura como el de Fondazione Eni Enrico Mattei para decidir la mejor forma para reducir el impacto en el cambio del clima global (56). Es usado algunas veces para procedimientos muy específicos en situaciones particulares, tales como la clasificación de edificios por importancia histórica (57).Recientemente se ha aplicado a un proyecto que usa cámaras de vídeo para evaluar el estado de las carreteras en Virginia. Los ingenieros de carretas primero lo usaron para determinar el alcance optimo del proyecto, luego para justificar su presupuesto a los legisladores (58). Existe un trabajo sobre el BPL (29) desarrollado con el objetivo de analizar el comportamiento de esta técnica con respecto a otras, para describen una herramienta desarrollada llamada Dutch. El VOP es un marco desarrollado por Technology Builders Incorporated (TBI) con el objetivo de continuar el desarrollo y mejoramiento de la aplicación CaliberRM™. Karlsson, Thelin, Regnell, Berander and Wohlin (59) hacen alusión a una aplicación llamada Tool-Supported PWC que facilita el uso de la técnica de comparación de pares de requisitos. También es posible usar una simple hoja de cálculo para asignar prioridades como la Matriz de Wiegers (60) en la que se ponderan los requisitos atendiendo a 4 criterios: penalidad, riesgo, costo y beneficio. Muchos artículos mencionan la automatización del proceso de priorización como el Enfoque Costo-Valor para la priorización de requisitos (37), el método de optimización multiobjetivo por Zhang, Harman y Lim (47), el AHP y PG (40), el VOP (42), entre otros. Son herramientas no comercializadas, explican en qué consiste las técnicas pero no proporcionan la herramienta. De la bibliografía revisada no se tiene registro de alguna herramienta disponible que contenga un conjunto de técnicas, métodos y procesos para permitir al usuario escoger entre una de ellas según las características particulares del proceso en el que prioriza. Es por eso que se desea desarrollar una herramienta informática que brinde un conjunto de opciones.

1.8 Metodologías de Desarrollo de Software

Para realizar una correcta implementación de la aplicación es necesario regirse por metodologías de desarrollo del software que permitan organizar el proceso de desarrollo de la aplicación. Existen muchas metodologías tales como RUP y XP. Escoger entre una de ellas implica el estudio de cada una de ellas teniendo en cuenta las condiciones, circunstancias con que se desarrolle la aplicación así como las necesidades de los stakeholders.

(RUP) Rational Unified Process

El colectivo de autores de “Rational Unified Process” (61) alegan que RUP es un marco de trabajo genérico con la facilidad de especializarse para una variedad de procesos de desarrollo de software. Está basado en componentes, está dirigido por caso de usos. Dividen a esta metodología 4 fases:

I. Inicio, El Objetivo en esta etapa es determinar la visión del proyecto.
II. Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.
III. Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
IV. Transmisión, El objetivo es llegar a obtener el release del proyecto.

RUP es un proceso interdisciplinario que se caracteriza por ser incremental. Permite evaluar tempranamente los riesgos (62). Coincidiendo con Castro Domínguez (62), RUP es más apropiada para proyectos grandes, requiere de personal para liberar la cantidad de artefactos que exige.

SCRUM

Esta metodología define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto (63). SCRUM está dirigida a la gestión de procesos de desarrollo de software (64).

Microsoft Solution Framework (MSF)

Se asegura en (65) que esta metodología es un enfoque personalizable. El artículo publicado en (66) coincide también con que es flexible que se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. Sin embargo esta se extiende mucho, prácticamente hay que documentarlo todo (66) , (65).

XP (Extreme Programming)

Es una metodología ágil. Es una de las metodologías más exitosas actualmente (66). Muy útiles para proyectos de corto plazo y personal. Consiste en una programación extrema por lo que los especialistas deben estar altamente calificados. Mendoza aclara que una de las características de XP es tener como

parte del equipo al cliente o usuario final por lo que la comunicación entre los stakeholders es muy importante. Esta metodología se enfoca en la programación, disminuyendo la generación de productos de artefactos. Esta metodología depende de la buena comunicación entre los stakeholders, que en caso de dañarse, podría paralizar el desarrollo del software.

Atendiendo a las necesidades y condiciones de la investigación donde realizar una aplicación correctamente ejecutable, y que se realice a corto plazo, XP encajaría perfectamente en la elección en cuanto a metodología de desarrollo se trata.

Las características fundamentales de esta metodología ágil son:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas.
- Programación en parejas.
- Frecuente integración del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código.
- Propiedad del código compartida.
- Simplicidad en el código.

Hurtado, Bastiarrica (67) y Canós, Letelier, y Penadés (68) determinan 6 etapas de XP:

1. *Exploración:* Constituye la primera fase donde los clientes a grandes rasgos plantean las historias usuarios. El equipo de especialistas se familiariza con las herramientas, tecnologías a utilizar.
2. *Planificación:* Se priorizan las historias de usuario utilizando el planning game como herramienta. Los programadores estiman cuánto esfuerzo requiere cada historia y a partir de allí se define el cronograma. El cliente decide las historias que se seleccionarán para cada iteración.

3. *Primera iteración*: Crea un sistema con la arquitectura del sistema completo con las historias de usuarios que harán cumplir la construcción de la estructura para el sistema completo.
4. *Producción:* Requiere prueba y comprobación extra del funcionamiento del sistema antes de que éste se pueda liberar al cliente. En esta fase, los nuevos cambios pueden todavía ser encontrados y debe tomarse la decisión de si se incluyen o no en el release actual. Las ideas y las sugerencias pospuestas se documentan para una puesta en práctica posterior por ejemplo en la fase de mantenimiento.
5. *Mantenimiento:* Requiere de un mayor esfuerzo para satisfacer también las tareas del cliente. Así, la velocidad del desarrollo puede desacelerar después de que el sistema esté en la producción. La fase de mantenimiento puede requerir la incorporación de nueva gente y cambiar la estructura del equipo.
6. *Muerte:* Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Se escoge la metodología XP dada sus características que permiten enfocarse a la programación de la aplicación y no al diseño, favoreciendo a la velocidad de desarrollo de la aplicación.

1.9 Lenguaje de Programación

C

Kernighan y Ritchie (69) describen a C como un lenguaje de programación de propósito general. Es muy útil y su aprendizaje a un nivel básico es relativamente sencillo y no requiere de mucho tiempo. Tiene un repertorio de instrucciones básicas relativamente pequeño. Aunque incluye numerosas funciones de biblioteca que mejoran las instrucciones básicas. Además los usuarios pueden escribir bibliotecas adicionales para su propio uso. Es uno de los más usados para la programación de algoritmos genéticos.

C++

Es un lenguaje de programación creado por Bjarne Stroustrup en los años 80. Es una variante de C para la manipulación de objetos. Es un lenguaje imperativo orientado a objetos muy ligado al hardware

subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero con elementos que le permiten también un estilo de programación con alto nivel de abstracción. “No es un lenguaje orientado a objetos puro (en el sentido en que puede serlo Java por ejemplo), además no nació como un ejercicio académico de diseño. Se trata simplemente del sucesor de un lenguaje de programación hecho por programadores (de alto nivel) para programadores, lo que se traduce en un diseño pragmático al que se le han ido añadiendo todos los elementos que la práctica aconsejaba como necesarios, con independencia de su belleza o purismo conceptual ha experimentado un extraordinario éxito desde su creación. De hecho, muchos sistemas operativos, compiladores e intérpretes han sido escritos en C++. Es un lenguaje de nivel intermedio, pudiéndose utilizar tanto para escribir software de bajo nivel, como drivers y componentes de sistemas operativos, como para el desarrollo rápido de aplicaciones.

C#

Es un lenguaje de programación orientado a objetos creado por Microsoft como parte de la iniciativa .NET FrameWork y escrito por Andres Hejlsberg en 1999. La última versión conocida es la 3.0 que salió a la luz el 17 de noviembre del 2007, propiedad de la compañía Microsoft. En su constitución, a diferencia de lenguajes como C / C++, que proporcionan un altísimo grado de control de los procesos permitiendo el uso de punteros y muchas otras funciones de bajo nivel, y otros incluidos dentro de este entorno .NET, como Microsoft Visual Basic, que posee un alto nivel y más facilidad a la hora de desarrollar una aplicación, C#, se encuentra en un término intermedio ya que ayuda a desarrollar aplicaciones rápidas pero que también permiten un gran control e integración con el desarrollo de aplicaciones Web, XML, y muchas otras de las tecnologías recientes. Este lenguaje no genera código nativo y para ejecutar los programas que se realicen, la computadora tiene que tener instalado .NET. Tiene que cargar su CLR (Common Language Infrastructure) y el Framework. NET cada vez que se ejecuta un programa en C # la carga del mismo es considerablemente mayor que la carga de un programa equivalente en C ++.

Java

El lenguaje de programación Java fue desarrollado por Sun Microsystems a principios de los 90’s con la idea original de usarlo para la creación de páginas WEB. Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de tal forma que prácticamente todo el Java de Sun es software libre. Según los datos recopilados por el Índice Comunitario de Programación del 2011 (70), Java es el lenguaje más popular dentro de la comunidad

internacional. Es Orientado a objetos, agrupa en estructuras encapsuladas tanto sus datos como las funciones que manipulan esos datos. Es interpretado, ya que los bytecodes, un formato intermedio indiferente a la arquitectura, se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real**.** Fue diseñado para crear software altamente fiable, proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una gran cantidad de errores (la aritmética de punteros). Java soporta aplicaciones que serán ejecutadas en variados entornos de red, diseñado para transportar el código eficientemente a múltiples plataformas de hardware y software. La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras. Toma mucha de su sintaxis de C y C++ pero tiene un modelo de objetos más simple eliminando herramientas como la manipulación directa de punteros o memoria. Se escoge el lenguaje Java porque tiene muchas ventajas. Ha sido diseñado de modo de eliminar las complejidades de otros lenguajes como C y C++. Es gratuito, y no hay que pagar la licencia para ".NET" como C#. Así mismo es un lenguaje muy completo y poderoso, se pueden realizar muchas tareas con él, pues posee una librería y utilidades muy completas que facilitan la programación. Existen trabajos (71), (72), (73) en el que emplean java para algoritmos genéticos. No hay sobrecarga de operadores en comparación con C++.

1.10 IDE[11] para Java

Eclipse: Es uno de los entornos Java más utilizados a nivel profesional. El paquete básico de Eclipse se puede expandir mediante la instalación de plugins para añadir funcionalidades a medida que se vayan necesitando.

JCreator: software comercial. Este IDE está escrito en C++ y omite herramientas para desarrollos gráficos, lo cual lo hace más rápido y eficiente que otros IDEs. Expandible mediante plugins

BlueJ: Es un entorno de desarrollo dirigido al aprendizaje de Java (entorno académico) y sin uso a nivel profesional. Es utilizado en distintas universidades para la enseñanza de Java. Destaca por ser sencillo e incluir algunas funcionalidades dirigidas a que las personas que estén aprendiendo tengan mayor facilidad para comprender aspectos clave de la programación orientada a objetos.

[11] Ambiente de Desarrollo Integrado

NetBeans: Otro de los entornos Java muy utilizados, también expandible mediante plugins. Facilita bastante el diseño gráfico asociado a aplicaciones Java.

Se escoge el Neatbeans pues la plataforma puede ser usada para desarrollar cualquier tipo de aplicación. Se pueden reutilizar los módulos, es de fácil instalación y actualización. Además de que el BlueJ es para el aprendizaje, el JCreator es comercial y en comparación con el Eclipse el Neatbeans tiene mayores herramientas como los swings que vienen implementados y los que se pueden tener acceso por librerías.

1.11 Conclusiones parciales

En este capítulo se realizó un análisis sobre los conceptos y definiciones que rodean el proceso de priorización de requisitos, así como el estado del arte de las diferentes técnicas, procesos, métodos más utilizados para ello en el mundo así como herramientas informáticas desarrolladas. Lo que contribuyó a la selección y adaptación de diferentes conceptos y técnicas atendiendo al problema planteado tales como el método de optimización multiobjetivo al emplear el AHP para la comparación entre los requisitos entre varios stakeholders analizando desde varios criterios. Para la implementación de la aplicación desktop fue necesario que se establecieran comparaciones no solo enfocadas a las características técnicas sino de acuerdo a la factibilidad en cuanto a recursos y tiempo para la asimilación de las tecnologías en cuestión escogiendo el lenguaje Java y el IDE Neatbeans. Se justificó la selección de la metodología XP para desarrollar dicha aplicación.

Capítulo 2. Propuesta de solución

2.1 Introducción

En este capítulo se describe cómo se realiza el proceso de priorización de requisitos en la UCI, las técnicas o métodos que emplean para dicho proceso así como los factores o aspectos que determinan para seleccionar los requisitos. Se presenta la propuesta del sistema. Se documenta la primera y segunda fase de la metodología.

2.2 Proceso de Priorización de Requisitos de Software en la UCI

Se realizó una entrevista (anexo 1) a proyectos de la UCI para saber cómo realizan la priorización. De forma general, en muchos proyectos establecen prioridades de forma empírica según la apreciación de los especialistas de software. En otros, analizan los casos de uso separándolos en tres grupos de prioridad: alta, media y, baja. Esta prioridad es calculada con el análisis de 4 criterios: la criticidad que se refiere a la importancia desde el punto de vista del negocio, la dependencia entre los requisitos, la estabilidad el caso de uso ante los procesos de cambio que ocurren en la organización, y la frecuencia que ayuda a identificar los casos de uso que mayor nivel de ocurrencia tienen y por lo tanto se desean tener implementados con anterioridad. Esta prioridad se compara con la complejidad de los casos de uso, que se clasifican bajos los mismos conceptos de alta, media y baja. Este método se basa en los principios de la Asignación Numérica, técnica caracterizada por su sencillez. Sin embargo, cuando se tienen muchos requisitos funcionales, desglosarlos en casos de uso y analizar su prioridad de cada uno de ellos, dificulta la toma de decisiones. Clasificar los casos de uso por cada criterio en altos, medios o bajos, no ofrece la información necesaria para discernir entre cada requisito.

2.3 Información que se maneja

Información perteneciente a la Ingeniería de Requisitos, aspectos relacionados sobre los requisitos extraídos en la fase de elicitación, sobre el análisis y posterior negociación entre las partes interesadas en la fase de análisis. Constituye los primeros pasos del desarrollo de la aplicación.

2.4 Propuesta de sistema

Con el objetivo de facilitar el proceso de decidir cuáles son los requisitos, funcionalidades, características que se desarrollarán en las siguientes fases de desarrollo del software, se decide crear una herramienta de apoyo a la toma de decisiones en la UCI. Esta herramienta debe reunir un conjunto de las técnicas, métodos y procesos más comunes y con procedimientos diferentes proporcionándole a los especialistas escoger la que más se ajuste a sus necesidades y objetivos. Esta herramienta de apoyo será una aplicación desktop llamada GESTREQ.

Después de estudiadas en el capítulo anterior todas las técnicas, métodos, y procesos, se decidió implementar los que presentan características particulares, diferentes y que a su vez arrojen resultados satisfactorios. Se conformará entonces una aplicación con diversas opciones para el usuario donde pueda escoger, a su consideración, la mejor a emplear. En la tabla 5 se muestra un resumen de las técnicas, métodos y procesos analizados en el capítulo para una mejor comprensión.

Nombre del enfoque	**Nivel del enfoque**	**Semejanza con otros enfoques**		**Tipos de Requisitos que se pueden ser priorizados**	**Criterios de selección que tienen en cuenta**
		Información que brinda	**Procedimiento**		
Asignación Numérica.	Técnica. Escala Nominal.	MOSCOW.	MOSCOW, PG.	Requisitos Funcionales y no Funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio, en este caso es el valor de importancia.
MOSCOW.	Técnica. Escala Nominal.	Asignación Numérica.	Asignación Numérica, PG.	Requisitos Funcionales y no Funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio, en este caso es el valor de importancia.
BPL.	Técnica. Escala Ordinal.	BST, Burbuja, PG, Priority Groups.	BST, Burbuja, AHP, MST, Jerarquía AHP, AHP combinado con el PG.	No es recomendable priorizar los funcionales y no funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. En ningún estudio de los analizados, explican cómo calculan la prioridad. Quedaría por parte de los stakeholders definir el criterio de selección.
BST.	Técnica.	BPL, Burbuja	BPL, Burbuja,	No es recomendable	Es una técnica por lo que

	Escala Ordinal.	PG, Priority Groups.	AHP, MST, Jerarquía AHP, AHP combinado con el PG.	priorizar los funcionales y no funcionales juntos.	tienen en cuenta un solo criterio. En ningún estudio de los analizados, explican cómo calculan la prioridad. Quedaría por parte de los stakeholders definir el criterio de selección.
Burbuja.	Técnica. Escala Ordinal.	BPL, BST PG, Priority Groups.	BPL, BST, AHP, MST, Jerarquía AHP, AHP combinado con el PG.	No es recomendable priorizar los funcionales y no funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. En ningún estudio de los analizados, explican cómo calculan la prioridad. Quedaría por parte de los stakeholders definir el criterio de selección.
Priority Groups.	Técnica. Escala Ordinal.	BPL, BST PG, Burbuja.	Ninguna.	No es recomendable priorizar los funcionales y no funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. En ningún estudio de los analizados, explican cómo calculan la prioridad. Quedaría por parte de los stakeholders definir el criterio de selección.
PG.	Técnica. Escala Ordinal.	BPL, BST Priority Groups, Burbuja.	MOSCOW, Asignación Numérica.	Requisitos Funcionales y no Funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. En esta investigación se concluyó que la prioridad es determinada a partir del grado de importancia que para el cliente representa.
100 puntos.	Técnica. Escala Ratio.	AHP, MST, Jerarquía AHP, AHP combinado con el PG, HCV.	HCV.	Requisitos Funcionales y no Funcionales juntos	Es una técnica por lo que tienen en cuenta un solo criterio. Para asignar prioridades analizan el grado de importancia de los requisitos.
HCV.	Técnica. Escala	AHP, MST, Jerarquía	100 puntos.	Requisitos Funcionales y no Funcionales	Es una técnica por lo que tienen en cuenta un solo

	Ratio.	AHP, AHP combinado con el PG, 100 puntos.		juntos.	criterio. Para asignar prioridades analizan el grado de importancia de los requisitos.
AHP.	Técnica. Escala Ratio.	HCV, MST, Jerarquía AHP, AHP combinado con el PG, 100 puntos.	MST, Jerarquía AHP, AHP combinado con el PG, BPL, BST, Burbuja.	No es recomendable priorizar los funcionales y no funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. Para asignar prioridades analizan el grado de importancia de los requisitos.
Planning Game con el AHP.	Técnica. Escala Ratio.	HCV, MST, Jerarquía AHP, AHP, PG, 100 puntos.	HCV, MST, Jerarquía AHP, AHP, PG.	Divide los requisitos en grupos de importancia por lo que con una correcta división de estos requisitos se pueden comprar los requisitos funcionales y no funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. Para asignar prioridades analizan el grado de importancia de los requisitos.
Jerarquía AHP	Técnica. Escala Ratio	HCV, MST, AHP, AHP combinado con el PG, 100 puntos.	MST, AHP, AHP combinado con el PG, BPL, BST, Burbuja.	Divide los requisitos en grupos jerárquicos por lo que con una correcta división de estos requisitos se pueden comprar los requisitos funcionales y no funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. .Para asignar prioridades analizan el grado de importancia de los requisitos.
MST	Técnica. Escala Ratio	HCV, Jerarquía AHP, AHP, AHP combinado con el PG, 100 puntos.	Jerarquía AHP, AHP, AHP combinado con el PG, BPL, BST, Burbuja.	No es recomendable priorizar los funcionales y no funcionales juntos.	Es una técnica por lo que tienen en cuenta un solo criterio. Para asignar prioridades analizan el grado de importancia de los requisitos.
Matriz de Wiegers.	Método.	VOP	VOP	Requisitos Funcionales y no Funcionales juntos	Se predefinen 4 criterios: penalidad, costo, riesgo e importancia.

Enfoque Costo-Valor para la priorización de requisitos.	Método.	Jerarquía AHP, AHP, AHP combinado con el PG, HCV, 100 puntos, MST.	Jerarquía AHP, AHP, AHP combinado con el PG, MST.	No es recomendable priorizar los funcionales y no funcionales juntos.	Se predefinen 2 criterios: costo e importancia.
VOP.	Método.	Matriz de Wiegers.	Matriz de Wiegers.	Requisitos Funcionales y no Funcionales juntos.	Se predefinen 4 criterios: penalidad, costo, riesgo e importancia. Además de los subcriterios de la importancia que pueden los stakeholders definir.
Proceso de Optimización Multiobjetivo.	Proceso.	BST, Burbuja, PG, Priority Groups, BPL.	Ninguna.	No es recomendable priorizar los funcionales y no funcionales juntos.	Se predefinen 3criterios: costo, importancia, y dependencias entre los requisitos.

Tabla 5 Resumen de las técnicas, métodos y procesos estudiados.

Con este breve análisis se implementarán las técnicas, métodos y procesos descritas excepto el MST, árbol de expansión mínima, pues es una modificación del AHP que elimina una de las características notorias de este: la redundancia. El Árbol binario de búsqueda (BST) y el BPL se eliminaron de las técnicas escogidas, pues presentan el mismo comportamiento de selección que la Técnica Burbuja y esta presenta mejores resultados (2). Se eliminó el MoSCoW pues se basa en el mismo concepto de asignación numérica y este último es más conocido. Producto a la similitud entre el VOP y la Matriz Wiegers en cuanto a la forma en como realizan la priorización, se decidió implementar una híbrido que le permita al usuario adicionar, dentro de los criterios definidos por Wiegers, subcriterios que granulen dichos criterios. A pesar de la particularidad del Priority Groups en cómo realiza la priorización, arroja los peores resultados para cualquier cantidad de requisitos, según los resultados arrojados por Vestola (2), Karsslon, Wohlin, Regnell (30) por lo que se decidió no implementarlo. En el caso del Enfoque Costo-Valor para la priorización de requisitos, se agrega al Método de Optimización Multiobjetivo. Es decir, los valores que necesita el Método de Optimización Multiobjetivo de cada requisito, costo-valor, será proporcionado por el

modelo del Enfoque Costo-Valor. Esta decisión es basada en los análisis del capítulo 1 sobre la poca facilidad de uso la ubicación a la hora de decidir el conjunto de requisitos de la gráfica del par (costo, valor), que además no garantiza la selección de las soluciones aceptables.

2.5 Requisitos No Funcionales

Usabilidad

El sistema a desarrollar debe prestar facilidades de uso que satisfagan las necesidades de todos los usuarios.

Portabilidad

El sistema será multiplataforma.

Apariencia o interfaz externa

La interfaz debe ser de fácil comprensión en su funcionamiento, permitiendo la utilización del sistema sin mucho entrenamiento, debe contar con interfaces uniformes y con los mismos colores y diseños. Se debe garantizar que los colores de la interfaz de la aplicación sean claros y debe presentar mensajes sin ambigüedades.

Requisitos de rendimiento

El sistema debe administrar eficiente y eficazmente los recursos dado que debe poder ejecutarse bajo diversas plataformas o sistemas operativos que administran de forma diferente los recursos.

2.6 Exploración

Constituye la primera fase de la metodología XP en donde el equipo de especialistas se familiariza con las herramientas, tecnologías a utilizar. Se realizan las historias de usuarios (HU), donde el cliente describe a grandes rasgos lo que él desea que tenga la aplicación. Se extrajeron (anexo 2) 14 Historias de Usuarios tales como:

Historia de Usuario 1 Administrar Requisitos

HISTORIA DE USUARIO	
Número: 1	Usuario: Especialista en desarrollo de software designado
Nombre de la Historia: Administrar requisitos	

Prioridad en negocio: alta (alta/media/baja)	Iteración asignada:1
Riesgo en desarrollo: medio (alta/media/baja)	Puntos estimados:1
Descripción: El usuario podrá adicionar, modificar, eliminar los requisitos con sus criterios de priorización, atendiendo a la técnica, método o proceso escogido por el usuario para la priorización	
Observaciones:	

Tabla 6 Descripción de la HU 1 Administrar requisitos

2.7 Planificación

La planificación es la fase donde el cliente y el grupo de desarrollo se reúnen y acuerdan el orden en el que se implementarán las historias de usuario y asociadas a éstas, las entregas. Esta fase consiste principalmente en una o varias reuniones grupales de planificación. Dando como resultado de esta fase, el Plan de Entregas (74). Se establece la prioridad de cada historia de usuario y seguidamente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas y se toman acuerdos sobre el contenido de la primera entrega y además se determina un cronograma en conjunto con el cliente.

2.7.2.1 Plan de Entrega

Después de tener definidas las historias de usuarios, se realiza el Plan de Entregas con la intención de que los programadores obtengan una estimación de dichas historias en cuanto al nivel de detalle y así fijar el período de tiempo que se puede tardar en la implementación de cada una.

Plan de Entrega

Historias de Usuarios	**Iteración 1 (20 / 3 / 2013)**	**Iteración 2 (10 / 4 / 2013)**	**Iteración 3 (1 / 5/ 2013)**
Administrar Requisitos. Administrar Criterios. Administrar Expertos. Manipular ficheros.	3	Terminado	Terminado

Exportar documento con información. Utilizar Técnicas AHP y Jerarquía AHP. Utilizar el Planning Game.			
Utilizar el método Estrategia Cognitiva. Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC) . Utilizar el Método Unión Matriz Wiegers-VOP. Utilizar Técnica Planning Combinado con el AHP.	No empezado	3	Terminado
Utilizar la Técnica Burbuja. Utilizar El Proceso de Optimización Multiobjetivo.	No empezado	No empezado	3

Tabla 8 Plan de Entrega

A continuación se muestran algunos prototipo de interfaz:

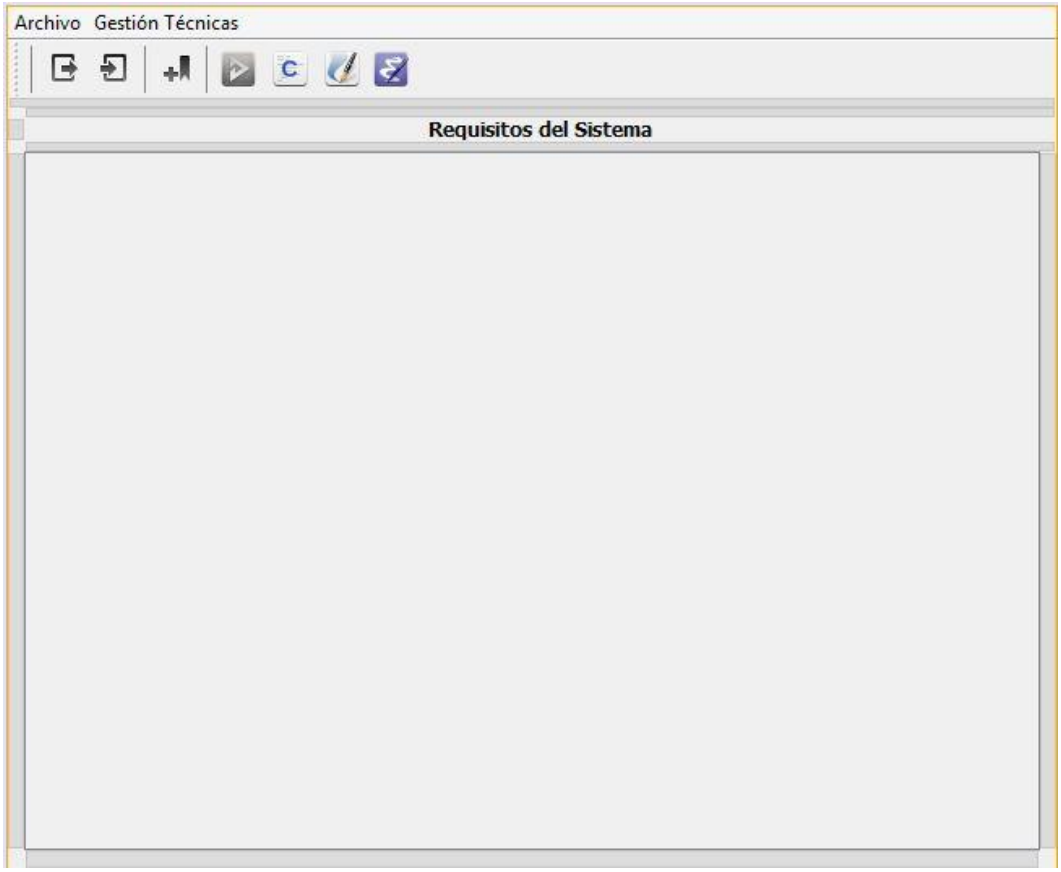

Figura 5 Prototipo de Interfaz Menú Principal

2.7.2 Arquitectura del Sistema

La aplicación presenta una arquitectura de N Capas pues permite organizar el modelo de diseño de forma que puedan estar físicamente distribuidas y así simplificar la comprensión y la organización de sistemas complejos (75), para la solución de la aplicación se definieron las siguientes capas, la capa Negocio encargada de modelar toda la lógica de negocio que estará compuesta por otras tres capas donde aparecen entidades y funcionalidades que servirán para dar solución al sistema, y la capa Vistas que contiene todas las interfaces que interactúan con todas las clases del sistema.

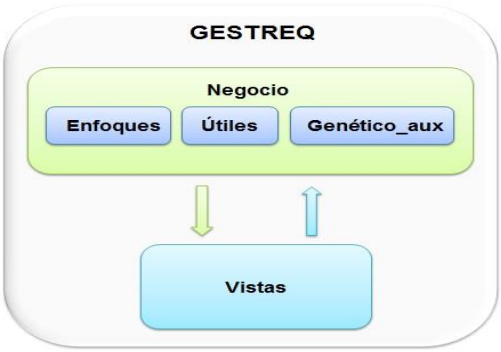

Figura 8: Arquitectura del Sistema

2.7.3 Patrones de diseño

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. Son principios generales de soluciones que aplican ciertos estilos y ayudan a la creación del software. Es una descripción de un problema y la solución a la que le da el nombre y que se puede aplicar en nuevos contextos. Muchos patrones ayudan a asignar responsabilidades a los objetos (76).

2.7.3.1 Patrones creacionales

Permiten que el sistema sea independiente de cómo se crean, componen o representan sus objetos.

Builder

Qué solución propone:

Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

Cuál es el problema a solucionar:

Se necesita controlar la construcción de los diferentes objetos. Este patrón va a ser muy útil para la reutilización de la implementación de técnicas que forman parte de otras como el AHP.

2.7.3.2 Patrones GRASP

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar (grasping) estos principios.

Experto

Qué solución propone:

Asignar una responsabilidad al experto en información -la clase que tiene la información necesaria para realizar la responsabilidad -.

Cuál es el problema a solucionar:

Cuando se definen las interacciones entre los objetos, tomamos decisiones sobre la asignación de responsabilidades a las clases software. Si se hace bien, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y existen más oportunidades para reutilizar componentes en futuras aplicaciones. Este patrón se utilizaría para decidir las clases que tendrían la responsabilidad de implementar las técnica, métodos y procesos, las cuales tendrían la información referente al procedimiento de cada uno.

Creador

Qué solución propone:

Asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes:

- B agrega objetos de A.
- B contiene objetos de A.
- B registra instancias de objetos de A.
- B utiliza más estrechamente objetos de A.
- B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A).

B es un creador de los objetos A.

Cuál es el problema a solucionar:

Dar la responsabilidad de la creación de una nueva instancia de alguna clase. Se utilizaría para definir una clase controladora llamada GESTREQ que a partir de su manipulación se decidiría cuáles objetos crear o no en situaciones determinadas.

Alta cohesión

Qué solución propone:

Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Cuál es el problema a solucionar:

La necesidad de distribuir las responsabilidades y así equilibrar la carga de las clases. Existirían clases para cada uno de las técnicas, métodos y procesos que tendrían la responsabilidad de gestionar las informaciones de cada uno.

2.8 Conclusiones

En este capítulo se describió el proceso de priorización de los requisitos que se realiza en la UCI. Se describió la solución, señalando requisitos no funcionales del sistema, la arquitectura propuesta y los patrones de diseño. Se comenzó con la etapa de Exploración detallando las historias de usuarios por parte del cliente, estimando el tiempo de desarrollo de las mismas, se extrajeron 14 historias de usuarios. Se documentó la segunda etapa de la metodología XP donde se planteó el Plan de iteraciones, definiéndose tres iteraciones que agrupan las historias de usuarios identificadas y se creó el Plan de entrega de versiones, donde se publicarán versiones del software de acuerdo con la iteración que se esté realizando.

Capítulo 3. Diseño e Implementación

3.1 Introducción

En este capítulo se detallan el diseño e implementación del software a partir de la realización de las tareas de la ingeniería que se obtuvieron de las historias de usuarios, se definen relaciones y responsabilidades entre clases.

3.2 Tareas de la ingeniería (TI)

A partir de aquí, el cliente conjuntamente con el programador detalla que se va a hacer con cada una de las historias de usuarios, dividiéndolas en tareas de la ingeniería. A partir de las 14 Historias de Usuarios identificadas en el capítulo 2, se extrajeron 23 Tareas de Ingeniería (ver Anexo 3) tales como:

Iteración 1

TAREA DE INGENIERÍA	
Número Tarea: 1	Historia de Usuario (no.1): Administrar Requisitos
Nombre de la Tarea: Insertar Requisito	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados:1
Fecha Inicio:20/3/2013	Fecha Fin: 21/3/2013
Programadores Responsables: Tania Pérez y Malena Pereda	
Descripción: se muestra una interfaz donde el usuario debe introducir los datos del requisito como el identificador, su nombre y su descripción .	

Tabla 10 Descripción de la TI Insertar Requisito

3.3 Conclusiones

En este capítulo se presentaron los elementos necesarios para implementar la aplicación: las tareas de ingenierías. Fue un trabajo en equipo entre cliente y desarrolladores. Es un paso importante pues se describe las funcionalidades obtenidas de las historias de usuarios en las diferentes etapas o iteraciones de la aplicación.

Capítulo 4. Validación

4.1 Introducción

XP pone la comprobación como el fundamento del desarrollo, con cada programador escribiendo pruebas cuando escriben su código de producción. Las pruebas se integran en el proceso de integración continua y construcción lo que rinde una plataforma altamente estable para el desarrollo futuro. En este capítulo se describirán la propuesta de validación para la aplicación GESTREQ.

4.2 Pruebas de Aceptación

Las pruebas de aceptación son pruebas de caja negra que son diseñadas por el cliente para cada historia de usuario, y su meta principal es asegurar que las funcionalidades del sistema cumplen con los objetivos propuestos. En efecto, las pruebas de aceptación corresponden a una especie de documento de requisitos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención. A continuación se exponen las pruebas de aceptación utilizadas para probar el software divididas en las iteraciones correspondientes a cada historia de usuario. Se extrajeron 29 pruebas de aceptación.

4.1.1 Iteración 1

Prueba de Aceptación 1 HU Administrar Requisitos

Caso de Prueba de Aceptación	
Código: HU1_p1	**Historia de Usuario (No.1):** Administrar requisitos
Nombre: Insertar un requisito con datos incorrectos	
Descripción: Probar que no se puede insertar un requisito con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben de existir errores en los datos de los requisitos.	

Entrada/ Pasos de ejecución: Requisitos con casillas en blanco o datos con errores. El sistema comprueba que los datos estén correcto o completos.
Resultado esperado: El sistema debe mostrar un mensaje de error.
Evaluación de la prueba: Satisfactoria

Tabla 11 Prueba de Aceptación HU1_p1

4.2 Conclusiones

En este capítulo se explicó la importancia que la metodología XP considera para la validación de los artefactos desarrollados. Se documentaron las pruebas de aceptación descritas por el cliente atendiendo a sus necesidades y expectativas.

Conclusiones Generales

Se concluye este trabajo de diploma dando cumplimiento al objetivo propuesto para la realización del mismo. A través de este documento se explica el Proceso de Priorización de Requisitos de Software, cómo en este se toman decisiones en cuanto a la selección de los requisitos que se desarrollarán, así como las principales técnicas, métodos y procesos que se utilizan.

En esta investigación se arrojaron los siguientes resultados importantes:

- Se seleccionaron y modificaron las técnicas, métodos y procesos a partir de 17 identificados a través de un análisis bibliográfico desarrollado teniendo en cuenta el procedimiento que realizan, los resultados que arrojan.

- Se desarrolló una herramienta que apoye la toma de decisiones durante el proceso de análisis y negociación de los requisitos de software a partir.

- Se sometió al software a pruebas de aceptación permitiendo medir el grado de satisfacción del cliente y el cumplimiento de las historias de usuarios.

Recomendaciones

Analizando el desarrollo de esta investigación y sus resultados, se recomienda a los interesados continuar el estudio y desarrollo de las técnicas, métodos y procesos que recojan mayor información sobre los stakeholders y los criterios para la optimización del proceso priorización de los requisitos de software. Se enfatiza el empleo de esta herramienta para tomar decisiones en los procesos de negociación entre los stakeholders así como el desarrollo de una próxima versión de la herramienta GESTREQ.

Referencias Bibliográficas

1. BAHAMONDE, Richard Rossel José Manuel. *Un Acercamiento a la Ingeniería de Requerimientos*. November 2003. Universidad Tècnica Federico Santa María.

2. MIKKO VESTOLA. *A Comparison of Nine Basic Techniques for Requirements Prioritization*.

3. VIGGO AHL. *An experimental comparison of five prioritization methods*. Master Teis. Sweden : School of Engineering Blekinge Institute of Technology, 2005.

4. CARLOS MARTÍN AZZOLINI. *Un Enfoque de Priorización de Requerimientos, a partir de la Segmentación de las Preferencias de los Stakeholders*. Maestría. Universidad Nacional de La Plata, 2011.

5. TOMA DE DECISIONES. [online]. [Accessed 2 May 2013]. Available from: http://www.csintranet.org/competenciaslaborales/index.php?option=com_content&view=article&id=163:toma-de-decisiones&catid=55:competencias

6. ING. KARINA SÁNCHEZ TAMAYO. *Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones. MEPROI* [online]. Maestría. La Habana, Cuba : Universidad de las Ciencias Informáticas, 2010. Available from: http://www.amazon.com/Evaluaci%C3%B3n-prioridad-proyectos-para-decisiones/dp/384847798X

7. La toma de decisiones. [online]. [Accessed 2 May 2013]. Available from: http://www.cop.es/colegiados/m-00451/tomadeciones.htm

8. LÓPEZ GEOMAR and MAESTRE MILDRED TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR AL TÍTULO DE: LICENCIADO EN GERENCIA DE RECURSOS HUMANOS MATURÍN, MARZO. *ANÁLISIS DE LA INFORMACIÓN ADMINISTRATIVA PARA LA TOMA DE DECISIONES TÁCTICAS*. de grado. MATURÍN, VENEZUELA : UNIVERSIDAD DE ORIENTE, VENEZUELA, 2005.

9. ALAIN ABRAN, JAMES W. MOORE, PIERRE BOURQUE, ROBERT DUPUIS and LEONARD L. TRIPP. *Guide to the Software Engineering Body of Knowledge* [online]. 2004 Version. Los Alamitos, California : Deborah Plummer, 2004. ISBN 0-7695-2330-7. Available from: http://computer.org

10. P. BERANDER, K. KHAN and L. LEHTOLA. Towards a research framework on requirements prioritization,. In : *SERPS06*. Sweden, 2006. p. 39–48.

11. NGO-THE and RUHE, G. Decision Support in Requirements Engineering. In : AYBÜKE AURUM and CLAES WOHLIN (eds.), *Engineering and Managing Software Requirements. Berlin, Germany : Springer Verlag, 2005. p. 267–286. ISBN 978-3-540-25043-2.*

12. *INGENIERO JORGE ALBERTO GALVES UNIVERSIDAD TECNOLÓGICA DE PEREIRA. RESUMEN: 4 CAPITULOS DE INGENIERIA DE REQUERIMIENTOS*. 10 October 2006. UNIVERSIDAD TECNOLÓGICA DE PEREIRA.

13. *Traductor Inglés-Español* [online]. [Accessed 16 May 2013]. Available from: http://traductor.babylon.com/ingles/a-espanol/

14. COMMITTEE OF THE COMPUTER SOCIETY OF THE IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. 1990. The Institute of Electrical and Electronics Ehgineers.

15. IAM SOMMERVILLE. *Ingeniería del Software*. Séptima. Pearson Educación, 2008. ISBN 84-7829-074-5.

16. PATRIK BERANDER BLEKINGE INSTITUTE OF TECHNOLOGY LICENTIATE SERIES NO 2004:12 ISBN 91-7295-052-8 2004-12-16 DEPARTMENT OF. *Prioritization of Stakeholder Needs in Software Engineering. Understanding and Evaluation*. for the degree of Licentiate of Technology in Software Engineering. Sweden : School of Engineering Blekinge Institute of Technology, 2004.

17. ROGER S PRESMAN. *Ingeniería de Software. Un enfoque práctico*. Quinta Edición. McGraw-Hill, 2006. ISBN 0-07-709677-0.

18. JOS´E MANUEL BAHAMONDE, Richard Rossel. *Un Acercamiento a la Ingenier´ıa de Requerimientos*.

19. AMADOR DURÁN TORO, Beatriz Bernárdez Jiménez. *Metodología para la Elicitación de Requisitos de Sistemas Software*. Técnico. Sevilla : Universidad de Sevilla, 2002.

20. TEAM. *CMMI Product. Version 1.2*. 2006. CMMI® for Development.

21. JOS´E MANUEL BAHAMONDE, Richard Rossel. *Un Acercamiento a la Ingeniería de Requerimientos*. November 2003. Universidad Tècnica Federico Santa María.

22. IEEE COMPUTER SOCIETY PROFESSIONAL PRACTICES COMMITTEE. *Guide to the Software Engineering Body of Knowledge*. 2004.

23. QIAO MA. *The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements: A Systematic Literature Review*. Master Tesis. Auckland University, 2009.

24. A. PERINI, F. RICCA, A. SUSI and C. BAZZANELLA. An empirical study to compare the accuracy of ahp and cbranking techniques for requirements prioritization,. In : *CERE '07*. Washington, DC, USA : IEEE Computer Society, 2007. p. 23–35.

25. P. BERANDER and P. JÖNSSON. Hierarchical cumulative voting (hcv) prioritization of requirements in hierarchies,. 2006. Vol. 16, p. 819–849.

26. P. BERANDER and M. SVAHNBERG. Evaluating two ways of calculating priorities in requirements hierarchies - an experiment on hierarchical cumulative voting,. 2009. Vol. 82, no. 5, p. 836–850.

27. MOTUPALLY, P. *Using Satisfaction Arguments and Rich Traceability in Requirement Priorization*. Thesis of Master of Computer and Information Sciences. Auckland, EEUU : Auckland University of Technology, 2008.

28. MoSCoW Prioritisation method. [online]. [Accessed 11 March 2013]. Available from: http://www.coleyconsulting.co.uk/moscow.htm

29. T. BEBENSEE, I. VAN DE WEERD and S. BRINKKEMPER. Binary priority list for prioritizing software requirements. 2010.

30. JOACHIM KARLSSON, CLAES WOHLIN and BJÖN REGNELL. An evaluation of methods for prioritizing software requirements. *0950-5849/98/$19.00*. 1998. Vol. 39, p. 9.

31. DAYANA CARIDAD TEJERA HERNÁNDEZ and LEIDY BÁRBARA SÁNCHEZ ECHEVARRÍA. *Ingeniería de Requisitos para el desarrollo del Sistema de Gestión de Inventario Almacén (SIGIA). Módulo Nomencladores*. La Habana : Universidad de las Ciencias Informáticas., 2007.

32. ARELIS TAMAYO CARVAJAL and ELISABETH HERNÁNDEZ ROSARIO. *Ingeniería de requisitos de una herramienta para la gestión de requisitos en los proyectos productivos que se desarrollan en la Universidad de las Ciencias Informáticas*. Teisis de Pregrado. La Habana : Universidad de las Ciencias Informáticas, 2009.

33. T.L. SAATY. The Analytic Hierarchy Process. 1980. P. pp. 19–28.

34. E. N. WEISS. USING THE ANALYTIC HIERARCHY PROCESS IN A DYNAMIC ENVIRONMENT. In : *Mathl Modelling,. Great Britain : Pergamon Journals Ltd, 1987. p. 211–216,. 0270-0255/87, 3-5,.*

35. KARL WIEGERS. Software Requirements. Second Edition. 2003.

36. JORGE DOORN, GRACIELA HADAD and GLADYS KAPLAN. *Facilitando la Asignación de Prioridades a los Requisitos*. [no date].

37. J. KARLSSON and KEVIN RYAN. A Cost-Value Approach for Prioritizing Requirements. *Norges Teknisk-Naturvitenskapelige Universitet*. October 1997.

38. HO-WON JUNG. *Optimizing Value and Cost in Requirements Analysis*. August 1998. IEEE Software.

39. KARLSSON, J. Software Requirements Prioritizing. *Proceedings of ICRE´96*. 1996.

40. LENA KARLSSON, PATRIK BERANDER, BJÖRN REGNELL and CLAES WOHLIN. Requirements Prioritisation: An Experiment on Exhaustive Pair-Wise Comparisons versus Planning Game Partitioning. In : *Proceedings 8th Conference on Empirical Assessment in Software Engineering*. Edinburgh, UK., 2004. 2004:12. ISBN 91-7295-052-8 2004-12-16.

41. First Things First: Prioritizing Requirements. [online]. [Accessed 29 April 2013]. Available from: http://www.processimpact.com/articles/prioritizing.html

42. JIM AZAR, RANDY K. SMITH and DAVID CORDES. Value-Oriented Prioritization: A Framework for Providing Visibility and Decision Support in the Requirements Engineering Process. .

43. NADINA MARTINEZ CAROD. *Priorización de Requerimientos de Software utilizando una estrategia cognitiva*.

44. A. OSYCZKA,. Multicriteria Optimization for Enginee- ring Design. *Academic Press,*. 1985.

45. ANTONIO J. NEBRO,, ENRIQUE ALBA and FRANCISCO LUNA. *Optimización Multi-Objetivo y Computación Grid*.

46. C. LI, J.M. VAN DEN AKKER and S. BRINKKEMPER. *An Integer Linear Programming Approach to Product Software Release Planning & Scheduling* [online]. Master Tesis. Utrecht, The Netherlands : Technical Report UU-CS. Department of Information and Computing Sciences Utrecht University, 2006. Available from: www.cs.uu.nl

47. YUANYUAN ZHANG, MARK HARMAN and SOO LING LIM. Empirical evaluation of search based requirements interaction management. [online]. 21 April 2012. Available from: www.elsevier.com/locate/infsof

48. D. GREER and G. RUHE. Software release planning: an evolutionary and iterative approach. [online]. 22 July 2003. Available from: www.elsevier.com/locate/infsof E-mail addresses: des.greer@qub.ac.uk (D. Greer); ruhe@ucalgary.ca (G.

49. YUANYUAN ZHANG, MARK HARMAN, ANTHONY FINKELSTEIN and S. AFSHIN MANSOURI. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation. [online]. 18 February 2011. DOI 10.1016/j.infsof.2011.02.001. Available from: www.elsevier.com/locate/infsof

50. GUALTIERO COLOMBO and CHRISTINE L. MUMFORD. *Comparing Algorithms, Representations and Operators for the Multi-Objective Knapsack Problem*. School of Computer Science Cardiff University United Kingdom.

51. GUALTIERO COLOMBO and CHRISTINE L. MUMFORD. *Comparing Algorithms, Representations and Operators for the Multi-Objective Knapsack Problem*.

52. CRINA GROSAN, MIHAI OLTEAN and D. DUMITRESCU. NEW EVOLUTIONARY ALGORITHM FOR THE MULTIOBJECTIVE 0/1 KNAPSACK PROBLEM. In : *International Conference on Theory and Applications of Mathematics and Informatics*. Alba Iulia, 2003.

53. RANDY L. HAUPT and SUE ELLEN HAUPT A. *PRACTICAL GENETIC ALGORITHMS*. SECOND EDITION. Canada : A JOHN WILEY & SONS, INC., PUBLICATION, 2004. ISBN 0-471-45565-2.

54. MAYA HRISTAKEVA and DIPTI SHRESTHA. *Solving the 0-1 Knapsack Problem with Genetic Algorithms*. Computer Science Department Simpson College, [no date].

55. FINLAY, P.N. *Introducing decision support systems*. Segunda ed. Oxford, UK Cambridge: NCC Blackwell : Blackwell Publishers, 1994.

56. M, Berrittella, A. CERTA, M. ENEA and P. ZITO. An Analytic Hierarchy Process for the Evaluation of Transport Policies to Reduce Climate Change Impacts. In : 2007.

57. LIPPERT, BARBARA C and STEPHEN F. WEBER. *HIST 1.0; Decision Support Software for Rating Buildings by Historic Significance*. October 1995. National Institute of Standards and Technology.

58. LARSON, CHARLES D.; and ERNEST H. FORMAN. Application of the Analytic Hierarchy Process to Select Project Scope for Videologging and Pavement Condition Data Collection. In : *86th Annual Meeting Compendium of Papers CD-ROM, Transportation Research Board of the National Academies*. January 2007.

59. LENA KARLSSON, THOMAS THELIN, BJÖRN REGNELL, PATRIK BERANDER and CLAES WOHLIN. Pair-Wise Comparisons versus Planning Game Partitioning – Experiments on Requirements Prioritisation Techniques. *3-33*. 2007. Vol. 12, no. 1, p. 3–33. DOI 10.1007/s10664-006-7240-4.

60. KARL E. WIEGERS. *Software Requirements*. Segunda. Microsoft Press, 2003. ISBN 0-7356-1879-8.

61. IVAR JACOBSON, GRADY BOOCH and JAMES RUMBAUGH. *Rational Unified Process*. Madrid : Pearson Educación, [no date]. ISBN 84-829-036-2.

62. Exposicion Rup. [online]. [Accessed 28 January 2013]. Available from: http://www.slideshare.net/aliciadelcoral/exposicion-rup

63. SCHWABER K.,, BEEDLE M., and MARTIN R.C. *Agile Software Development with SCRUM*. 2001. Prentice Hall.

64. Diferencias entre scrum y xp. [online]. [Accessed 29 January 2013]. Available from: http://www.slideshare.net/deborahgal/diferencias-entre-scrum-y-xp-12219336

65. Descripción general de Microsoft Solutions Framework (MSF). [online]. [Accessed 29 January 2013]. Available from: http://msdn.microsoft.com/es-es/library/jj161047.aspx

66. http://www.informatizate.net - Metodologías De Desarrollo De Software. [online]. [Accessed 28 January 2013]. Available from: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

67. HURTADO, JULIO ARIEL and BASTIARRICA CECILIA. *Proyecto SIMEP-SW*. 8 May 2005.

68. JOSÉ H. CANÓS, PATRICIO LETELIER and Mª CARMEN PENADÉS. *Metodologías Ágiles para el desarrollo de software*. Universidad Politécnica de Valencia.

69. BRIAN W KERNIGHAN and DENNIS M. RITCHIE. *The C Programming Language*. 2da. Prenticf Hall Software Series, [no date].

70. Ranking Lenguajes de Programación Marzo 2011. [online]. [Accessed 20 May 2013]. Available from: http://www.desarrolloweb.com/de_interes/ranking-lenguajes-programacion-marzo-2011-4994.html

71. ELISA GUARDO, HAROLD CASTRO, GERMÁN BRAVO and ANDRÉS L. MEDAGLIA. *Uso de Tecnología Grid en Algoritmos de Optimización Evolutiva*. Colombia : COMIT, Departamento de Ingenieria de Sistemas y Computacion. Centro de Optimizacion y Probabilidad Aplicada (COPA), Departamento de Ingenieria Industrial. Universidad de los Andes-Colombia, [no date].

72. Las Ideas en Software: Sistema Para Optimizar Funciones Matemáticas con Algoritmos Genéticos en Java. [online]. [Accessed 31 January 2013]. Available from: http://lasideasensoftware.blogspot.com/2012/05/proyecto-en-diplomado-java.html

73. Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA. [online]. [Accessed 31 January 2013]. Available from: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jgap

74. ING. JOSÉ JOSKOWICZ. Reglas y Prácticas en eXtreme Programming. [online]. [Accessed 12 March 2013]. Available from: www.uls.edu.sv/index.php?option=com...view...reglas...

75. CÉSAR DE LA TORRE, UNAI ZORRILLA, MIGUEL ÁNGEL RAMOS and JAVIER CALVARRO. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0*. 1ra. 2010.

76. MC JUAN CARLOS OLIVARES ROJAS,. *Ptrones de Diseño* [online]. 10 May 2007. Available from: http://antares.itmorelia.edu.mx/~jcolivar/courses/dp07b/patrones.pdf.

Bibliografía

1. BAHAMONDE, Richard Rossel José Manuel. *Un Acercamiento a la Ingeniería de Requerimientos*. November 2003. Universidad Tècnica Federico Santa María.

2. MIKKO VESTOLA. *A Comparison of Nine Basic Techniques for Requirements Prioritization.*

3. VIGGO AHL. *An experimental comparison of five prioritization methods*. Master Teis. Sweden : School of Engineering Blekinge Institute of Technology, 2005.

4. CARLOS MARTÍN AZZOLINI. *Un Enfoque de Priorización de Requerimientos, a partir de la Segmentación de las Preferencias de los Stakeholders*. Maestría. Universidad Nacional de La Plata, 2011.

5. TOMA DE DECISIONES. [online]. [Accessed 2 May 2013]. Available from: http://www.csintranet.org/competenciaslaborales/index.php?option=com_content&view=article&id=163:toma-de-decisiones&catid=55:competencias

6. ING. KARINA SÁNCHEZ TAMAYO. *Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones. MEPROI* [online]. Maestría. La Habana, Cuba : Universidad de las Ciencias Informáticas, 2010. Available from: http://www.amazon.com/Evaluaci%C3%B3n-prioridad-proyectos-para-decisiones/dp/384847798X

7. La toma de decisiones. [online]. [Accessed 2 May 2013]. Available from: http://www.cop.es/colegiados/m-00451/tomadeciones.htm

8. LÓPEZ GEOMAR and MAESTRE MILDRED TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR AL TÍTULO DE: LICENCIADO EN GERENCIA DE RECURSOS HUMANOS MATURÍN, MARZO. *ANÁLISIS DE LA INFORMACIÓN ADMINISTRATIVA PARA LA TOMA DE DECISIONES TÁCTICAS*. de grado. MATURÍN, VENEZUELA : UNIVERSIDAD DE ORIENTE, VENEZUELA, 2005.

9. ALAIN ABRAN, JAMES W. MOORE, PIERRE BOURQUE, ROBERT DUPUIS and LEONARD L. TRIPP. *Guide to the Software Engineering Body of Knowledge* [online]. 2004 Version. Los Alamitos, California : Deborah Plummer, 2004. ISBN 0-7695-2330-7. Available from: http://computer.org

10. P. BERANDER, K. KHAN and L. LEHTOLA. Towards a research framework on requirements prioritization,. In : *SERPS06*. Sweden, 2006. p. 39–48.

11. NGO-THE and RUHE, G. Decision Support in Requirements Engineering. In : AYBÜKE AURUM and CLAES WOHLIN (eds.), *Engineering and Managing Software Requirements. Berlin, Germany : Springer Verlag, 2005. p. 267–286. ISBN 978-3-540-25043-2.*

12. *INGENIERO JORGE ALBERTO GALVES UNIVERSIDAD TECNOLÓGICA DE PEREIRA. RESUMEN: 4 CAPITULOS DE INGENIERIA DE REQUERIMIENTOS*. 10 October 2006. UNIVERSIDAD TECNOLÓGICA DE PEREIRA.

13. *Traductor Inglés-Español* [online]. [Accessed 16 May 2013]. Available from: http://traductor.babylon.com/ingles/a-espanol/

14. COMMITTEE OF THE COMPUTER SOCIETY OF THE IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. 1990. The Institute of Electrical and Electronics Ehgineers.

15. IAM SOMMERVILLE. *Ingeniería del Software*. Séptima. Pearson Educación, 2008. ISBN 84-7829-074-5.

16. PATRIK BERANDER BLEKINGE INSTITUTE OF TECHNOLOGY LICENTIATE SERIES NO 2004:12 ISBN 91-7295-052-8 2004-12-16 DEPARTMENT OF. *Prioritization of Stakeholder Needs in Software Engineering. Understanding and Evaluation*. for the degree of Licentiate of Technology in Software Engineering. Sweden : School of Engineering Blekinge Institute of Technology, 2004.

17. ROGER S PRESMAN. *Ingeniería de Software. Un enfoque práctico*. Quinta Edición. McGraw-Hill, 2006. ISBN 0-07-709677-0.

18. JOS´E MANUEL BAHAMONDE, Richard Rossel. *Un Acercamiento a la Ingenier´ıa de Requerimientos*.

19. AMADOR DURÁN TORO, Beatriz Bernárdez Jiménez. *Metodología para la Elicitación de Requisitos de Sistemas Software*. Técnico. Sevilla : Universidad de Sevilla, 2002.

20. TEAM. *CMMI Product. Version 1.2*. 2006. CMMI® for Development.

21. JOS´E MANUEL BAHAMONDE, Richard Rossel. *Un Acercamiento a la Ingeniería de Requerimientos*. November 2003. Universidad Tècnica Federico Santa María.

22. IEEE COMPUTER SOCIETY PROFESSIONAL PRACTICES COMMITTEE. *Guide to the Software Engineering Body of Knowledge*. 2004.

23. QIAO MA. *The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements: A Systematic Literature Review*. Master Tesis. Auckland University, 2009.

24. A. PERINI, F. RICCA, A. SUSI and C. BAZZANELLA. An empirical study to compare the accuracy of ahp and cbranking techniques for requirements prioritization,. In : *CERE '07*. Washington, DC, USA : IEEE Computer Society, 2007. p. 23–35.

25. P. BERANDER and P. JÖNSSON. Hierarchical cumulative voting (hcv) prioritization of requirements in hierarchies,. 2006. Vol. 16, p. 819–849.

26. P. BERANDER and M. SVAHNBERG. Evaluating two ways of calculating priorities in requirements hierarchies - an experiment on hierarchical cumulative voting,. 2009. Vol. 82, no. 5, p. 836–850.

27. MOTUPALLY, P. *Using Satisfaction Arguments and Rich Traceability in Requirement Priorization*. Thesis of Master of Computer and Information Sciences. Auckland, EEUU : Auckland University of Technology, 2008.

28. MoSCoW Prioritisation method. [online]. [Accessed 11 March 2013]. Available from: http://www.coleyconsulting.co.uk/moscow.htm

29. T. BEBENSEE, I. VAN DE WEERD and S. BRINKKEMPER. Binary priority list for prioritizing software requirements. 2010.

30. JOACHIM KARLSSON, CLAES WOHLIN and BJÖN REGNELL. An evaluation of methods for prioritizing software requirements. *0950-5849/98/$19.00*. 1998. Vol. 39, p. 9.

31. DAYANA CARIDAD TEJERA HERNÁNDEZ and LEIDY BÁRBARA SÁNCHEZ ECHEVARRÍA. *Ingeniería de Requisitos para el desarrollo del Sistema de Gestión de Inventario Almacén (SIGIA). Módulo Nomencladores*. La Habana : Universidad de las Ciencias Informáticas., 2007.

32. ARELIS TAMAYO CARVAJAL and ELISABETH HERNÁNDEZ ROSARIO. *Ingeniería de requisitos de una herramienta para la gestión de requisitos en los proyectos productivos que se desarrollan en la Universidad de las Ciencias Informáticas*. Teisis de Pregrado. La Habana : Universidad de las Ciencias Informáticas, 2009.

33. T.L. SAATY. The Analytic Hierarchy Process. 1980. P. pp. 19–28.

34. E. N. WEISS. USING THE ANALYTIC HIERARCHY PROCESS IN A DYNAMIC ENVIRONMENT. In : *Mathl Modelling,. Great Britain : Pergamon Journals Ltd, 1987. p. 211–216,. 0270-0255/87, 3-5,.*

35. KARL WIEGERS. Software Requirements. Second Edition. 2003.

36. JORGE DOORN, GRACIELA HADAD and GLADYS KAPLAN. *Facilitando la Asignación de Prioridades a los Requisitos*. [no date].

37. J. KARLSSON and KEVIN RYAN. A Cost-Value Approach for Prioritizing Requirements. *Norges Teknisk-Naturvitenskapelige Universitet*. October 1997.

38. HO-WON JUNG. *Optimizing Value and Cost in Requirements Analysis*. August 1998. IEEE Software.

39. KARLSSON, J. Software Requirements Prioritizing. *Proceedings of ICRE´96*. 1996.

40. LENA KARLSSON, PATRIK BERANDER, BJÖRN REGNELL and CLAES WOHLIN. Requirements Prioritisation: An Experiment on Exhaustive Pair-Wise Comparisons versus Planning Game Partitioning. In : *Proceedings 8th Conference on Empirical Assessment in Software Engineering*. Edinburgh, UK., 2004. 2004:12. ISBN 91-7295-052-8 2004-12-16.

41. First Things First: Prioritizing Requirements. [online]. [Accessed 29 April 2013]. Available from: http://www.processimpact.com/articles/prioritizing.html

42. JIM AZAR, RANDY K. SMITH and DAVID CORDES. Value-Oriented Prioritization: A Framework for Providing Visibility and Decision Support in the Requirements Engineering Process. .

43. NADINA MARTINEZ CAROD. *Priorización de Requerimientos de Software utilizando una estrategia cognitiva*.

44. A. OSYCZKA,. Multicriteria Optimization for Enginee- ring Design. *Academic Press,*. 1985.

45. ANTONIO J. NEBRO,, ENRIQUE ALBA and FRANCISCO LUNA. *Optimización Multi-Objetivo y Computación Grid*.

46. C. LI, J.M. VAN DEN AKKER and S. BRINKKEMPER. *An Integer Linear Programming Approach to Product Software Release Planning & Scheduling* [online]. Master Tesis. Utrecht, The Netherlands : Technical Report UU-CS. Department of Information and Computing Sciences Utrecht University, 2006. Available from: www.cs.uu.nl

47. YUANYUAN ZHANG, MARK HARMAN and SOO LING LIM. Empirical evaluation of search based requirements interaction management. [online]. 21 April 2012. Available from: www.elsevier.com/locate/infsof

48. D. GREER and G. RUHE. Software release planning: an evolutionary and iterative approach. [online]. 22 July 2003. Available from: www.elsevier.com/locate/infsof E-mail addresses: des.greer@qub.ac.uk (D. Greer); ruhe@ucalgary.ca (G.

49. YUANYUAN ZHANG, MARK HARMAN, ANTHONY FINKELSTEIN and S. AFSHIN MANSOURI. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation. [online]. 18 February 2011. DOI 10.1016/j.infsof.2011.02.001. Available from: www.elsevier.com/locate/infsof

50. GUALTIERO COLOMBO and CHRISTINE L. MUMFORD. *Comparing Algorithms, Representations and Operators for the Multi-Objective Knapsack Problem*. School of Computer Science Cardiff University United Kingdom.

51. GUALTIERO COLOMBO and CHRISTINE L. MUMFORD. *Comparing Algorithms, Representations and Operators for the Multi-Objective Knapsack Problem*.

52. CRINA GROSAN, MIHAI OLTEAN and D. DUMITRESCU. NEW EVOLUTIONARY ALGORITHM FOR THE MULTIOBJECTIVE 0/1 KNAPSACK PROBLEM. In : *International Conference on Theory and Applications of Mathematics and Informatics*. Alba Iulia, 2003.

53. RANDY L. HAUPT and SUE ELLEN HAUPT A. *PRACTICAL GENETIC ALGORITHMS*. SECOND EDITION. Canada : A JOHN WILEY & SONS, INC., PUBLICATION, 2004. ISBN 0-471-45565-2.

54. MAYA HRISTAKEVA and DIPTI SHRESTHA. *Solving the 0-1 Knapsack Problem with Genetic Algorithms*. Computer Science Department Simpson College, [no date].

55. FINLAY, P.N. *Introducing decision support systems*. Segunda ed. Oxford, UK Cambridge: NCC Blackwell : Blackwell Publishers, 1994.

56. M, Berrittella, A. CERTA, M. ENEA and P. ZITO. An Analytic Hierarchy Process for the Evaluation of Transport Policies to Reduce Climate Change Impacts. In : 2007.

57. LIPPERT, BARBARA C and STEPHEN F. WEBER. *HIST 1.0; Decision Support Software for Rating Buildings by Historic Significance*. October 1995. National Institute of Standards and Technology.

58. LARSON, CHARLES D.; and ERNEST H. FORMAN. Application of the Analytic Hierarchy Process to Select Project Scope for Videologging and Pavement Condition Data Collection. In : *86th Annual Meeting Compendium of Papers CD-ROM, Transportation Research Board of the National Academies*. January 2007.

59. LENA KARLSSON, THOMAS THELIN, BJÖRN REGNELL, PATRIK BERANDER and CLAES WOHLIN. Pair-Wise Comparisons versus Planning Game Partitioning – Experiments on Requirements Prioritisation Techniques. *3-33*. 2007. Vol. 12, no. 1, p. 3–33. DOI 10.1007/s10664-006-7240-4.

60. KARL E. WIEGERS. *Software Requirements*. Segunda. Microsoft Press, 2003. ISBN 0-7356-1879-8.

61. IVAR JACOBSON, GRADY BOOCH and JAMES RUMBAUGH. *Rational Unified Process*. Madrid : Pearson Educación, [no date]. ISBN 84-829-036-2.

62. Exposicion Rup. [online]. [Accessed 28 January 2013]. Available from: http://www.slideshare.net/aliciadelcoral/exposicion-rup

63. SCHWABER K.,, BEEDLE M., and MARTIN R.C. *Agile Software Development with SCRUM*. 2001. Prentice Hall.

64. Diferencias entre scrum y xp. [online]. [Accessed 29 January 2013]. Available from: http://www.slideshare.net/deborahgal/diferencias-entre-scrum-y-xp-12219336

65. Descripción general de Microsoft Solutions Framework (MSF). [online]. [Accessed 29 January 2013]. Available from: http://msdn.microsoft.com/es-es/library/jj161047.aspx

66. http://www.informatizate.net - Metodologías De Desarrollo De Software. [online]. [Accessed 28 January 2013]. Available from: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

67. HURTADO, JULIO ARIEL and BASTIARRICA CECILIA. *Proyecto SIMEP-SW*. 8 May 2005.

68. JOSÉ H. CANÓS, PATRICIO LETELIER and Mª CARMEN PENADÉS. *Metodologías Ágiles para el desarrollo de software*. Universidad Politécnica de Valencia.

69. BRIAN W KERNIGHAN and DENNIS M. RITCHIE. *The C Programming Language*. 2da. Prenticf Hall Software Series, [no date].

70. Ranking Lenguajes de Programación Marzo 2011. [online]. [Accessed 20 May 2013]. Available from: http://www.desarrolloweb.com/de_interes/ranking-lenguajes-programacion-marzo-2011-4994.html

71. ELISA GUARDO, HAROLD CASTRO, GERMÁN BRAVO and ANDRÉS L. MEDAGLIA. *Uso de Tecnología Grid en Algoritmos de Optimización Evolutiva*. Colombia : COMIT, Departamento de Ingenieria de Sistemas y Computacion. Centro de Optimizacion y Probabilidad Aplicada (COPA), Departamento de Ingenieria Industrial. Universidad de los Andes-Colombia, [no date].

72. Las Ideas en Software: Sistema Para Optimizar Funciones Matemáticas con Algoritmos Genéticos en Java. [online]. [Accessed 31 January 2013]. Available from: http://lasideasensoftware.blogspot.com/2012/05/proyecto-en-diplomado-java.html

73. Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA. [online]. [Accessed 31 January 2013]. Available from: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jgap

74. ING. JOSÉ JOSKOWICZ. Reglas y Prácticas en eXtreme Programming. [online]. [Accessed 12 March 2013]. Available from: www.uls.edu.sv/index.php?option=com...view...reglas...

75. CÉSAR DE LA TORRE, UNAI ZORRILLA, MIGUEL ÁNGEL RAMOS and JAVIER CALVARRO. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0*. 1ra. 2010.

76. MC JUAN CARLOS OLIVARES ROJAS,. *Ptrones de Diseño* [online]. 10 May 2007. Available from: http://antares.itmorelia.edu.mx/~jcolivar/courses/dp07b/patrones.pdf.

Anexos

Anexo 1 Entrevista

Entrevista para obtener información acerca de cómo se realiza el proceso de priorización de requisitos en los proyectos de la UCI.

Datos del Encuestado

Nombre y Apellidos: _____

Ocupación Laboral: _____

Rol del proyecto: _____

Facultad: _____

Nombre del proyecto: _____

Cuestionario

1- ¿En el proyecto se utiliza algún método para la priorización de requisitos?

____Si ____No_____ No sé.

En caso negativo: ¿Cómo se realiza la priorización?

_____

_____

En caso positivo: ¿Puede decir el nombre del método?_____

1.1- ¿Cuáles son los aspectos o criterios que tienen en cuenta para calcular la prioridad?_____

_____

Anexo 2 Historias de Usuario

Historia de Usuario 4 Manipular ficheros

HISTORIA DE USUARIO	
Número:4	Usuario: Especialista en desarrollo de software designado
Nombre de la Historia: Manipular ficheros	
Prioridad en negocio: alta (alta/media/baja)	Iteración asignada:1
Riesgo en desarrollo: medio (alta/media/baja)	Puntos estimados:4
Descripción: El sistema debe permitir a los usuarios exportar e importar ficheros que guarden la informaciones de los requisitos y una vez importado el fichero permitir su modificación.	
Observaciones:	

Tabla 17 Descripción de la HU 4 Manipular ficheros

Historia de Usuario 5 Exportar documento con información

HISTORIA DE USUARIO	
Número:5	Usuario: Especialista en desarrollo de software designado
Nombre de la Historia: Exportar documento con información	
Prioridad en negocio: alta (alta/media/baja)	Iteración asignada:1
Riesgo en desarrollo: medio (alta/media/baja)	Puntos estimados:5
Descripción: El sistema debe permitir a los usuarios exportar las informaciones de la priorización de los requisitos en documentos con formato PDF, en adobe que no permita su modificación una vez exportado la información.	
Observaciones:	

Tabla 18 Descripción de la HU 5 Exportar documento con información

Historia de Usuario 6 Utilizar Técnicas AHP y Jerarquía AHP

HISTORIA DE USUARIO	
Número: 6	Usuario: El conjunto de expertos
Nombre de la Historia: Utilizar Técnicas AHP y Jerarquía AHP	
Prioridad en negocio: alta (alta/media/baja)	Iteración asignada: 1
Riesgo en desarrollo: alta (alta/media/baja)	Puntos estimados: 6
Descripción: Los usuarios podrán utilizar las técnicas de AHP o Jerarquía AHP según estimen conveniente.	
Observaciones:	

Tabla 19 Descripción de la HU 6 Utilizar Técnicas AHP y Jerarquía AHP

Historia de Usuario 7 Utilizar el Planning Game

HISTORIA DE USUARIO	
Número: 7	Usuario: cliente
Nombre de la Historia: Utilizar el Planning Game	
Prioridad en negocio: alta (alta/media/baja)	Iteración asignada:1
Riesgo en desarrollo: alta (alta/media/baja)	Puntos estimados: 7
Descripción: El usuario podrá emplear esta técnica distribuyendo en las tres pilas de prioridad para obtener la lista de requisitos priorizados.	
Observaciones:	

Tabla 20 Descripción de la HU 7 Utilizar el Planning Game

Iteración 2

Historia de Usuario 8 Utilizar la Estrategia Cognitiva

HISTORIA DE USUARIO	
Número: 8	Usuario: El conjunto de expertos
Nombre de la Historia: Utilizar la Estrategia Cognitiva	
Prioridad en negocio: media (alta/media/baja)	Iteración asignada:2
Riesgo en desarrollo: media (alta/media/baja)	Puntos estimados: 1
Descripción: El usuario podrá utilizar la estrategia cognitiva propuesta por Nadina para priorizar los requisitos.	
Observaciones:	

Tabla 21 Descripción de la HU 8 Utilizar la Estrategia Cognitiva

Historia de Usuario 9 Utilizar Técnica asignación numérica

HISTORIA DE USUARIO	
Número:9	Usuario: Especialista en desarrollo de software designado
Nombre de la Historia: Utilizar Técnica asignación numérica	
Prioridad en negocio: media (alta/media/baja)	Iteración asignada:2
Riesgo en desarrollo: media (alta/media/baja)	Puntos estimados:2
Descripción: El usuario podrá utilizar la técnica de asignación numeral donde podrá separar cada requisito dado el valor numérico asignado.	

Tabla 22 Descripción de la HU 9 Utilizar Técnica asignación numérica

Historia de Usuario 10 Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC)

HISTORIA DE USUARIO	
Número: 10	Usuario: El conjunto de expertos
Nombre de la Historia: Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC)	
Prioridad en negocio: media (alta/media/baja)	Iteración asignada:2
Riesgo en desarrollo: baja (alta/media/baja)	Puntos estimados:3
Descripción: Los usuarios podrán utilizar la técnica de los 100 puntos para asignar puntos a cada requisito según sus criterios y así establecer sus respectivas prioridades. Reutilizando esta técnica podrán dividir en grupos jerárquicos los requisitos y emplear el HCV.	
Observaciones:	

Tabla 23 Descripción de la HU 10 Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC)

Historia de Usuario 11 Utilizar el Método Unión Matriz Wiegers-VOP

HISTORIA DE USUARIO	
Número: 11	Usuario: El especialista asignado
Nombre de la Historia: Utilizar el Método Unión Matriz Wiegers-VOP.	
Prioridad en negocio: media (alta/media/baja)	Iteración asignada:2
Riesgo en desarrollo: baja (alta/media/baja)	Puntos estimados:4
Descripción: Los usuarios podrán utilizar estableciendo prioridades a partir de los criterios: costo, penalidad, importancia, riesgos y desglosarlos en subcriterios que el usuario estime conveniente.	
Observaciones:	

Tabla 24 Descripción de la HU 11 Utilizar el Método Unión Matriz Wiegers-VOP

Historia de Usuario 12 Utilizar Técnica Planning Combinado con el AHP.

HISTORIA DE USUARIO	
Número: 12	Usuario: El especialista asignado
Nombre de la Historia: Utilizar Técnica Planning Combinado con el AHP.	
Prioridad en negocio: media (alta/media/baja)	Iteración asignada:2
Riesgo en desarrollo: baja (alta/media/baja)	Puntos estimados:5
Descripción: Los usuarios podrán emplear esta técnica combinando el AHP con el PG.	
Observaciones:	

Tabla 25 Descripción de la HU 12 Utilizar Técnica Planning Combinado con el AHP

Iteración 3

Historia de Usuario 13 Utilizar la Técnica Burbuja

HISTORIA DE USUARIO	
Número:13	Usuario: Especialista en desarrollo de software designado
Nombre de la Historia: Utilizar la técnica Burbuja	
Prioridad en negocio: baja (alta/media/baja)	Iteración asignada:3
Riesgo en desarrollo: baja (alta/media/baja)	Puntos estimados: 1
Descripción: El usuario podrá emplear esta técnica para obtener las prioridades de los requisitos, realizando comparaciones de dos en dos entre estos y dando como resultado una lista ordenada de los requisitos.	
Observaciones:	

Tabla 26 Descripción de la HU 13 Utilizar Técnica Planning Combinado con el AHP

Historia de Usuario 14 Utilizar el Proceso Optimización Multiobjetivo

HISTORIA DE USUARIO	
Número: 14	Usuario: El conjunto de expertos
Nombre de la Historia: Utilizar el Proceso Optimización Multiobjetivo	
Prioridad en negocio: alta (alta/media/baja)	Iteración asignada:3
Riesgo en desarrollo: alta (alta/media/baja)	Puntos estimados:2
Descripción: Los usuarios podrán emplear enfoque combinando el enfoque de costo-valor con el método de optimización multicriterio	
Observaciones:	

Tabla 27 Descripción de la HU 14 Utilizar El Método de Optimización Multiobjetivo

Anexo 3 Tareas de Ingenierías

Tarea de Ingeniería 4 Insertar Criterio

TAREA DE INGENIERÍA	
Número Tarea: 4	Historia de Usuario (no.2): Administrar Criterios
Nombre de la Tarea: Insertar Criterio	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados:0.4
Fecha Inicio:22/3/2013	Fecha Fin:23/3/2013

Programador Responsable: Tania Pérez
Descripción: Se muestra una interfaz donde el usuario debe introducir los datos del criterio como el su nombre y su descripción .

Tabla 28 Descripción de la TI Insertar Criterio

Tarea de Ingeniería 5 Modificar Criterio

TAREA DE INGENIERÍA	
Número Tarea: 5	Historia de Usuario(no.2): Administrar Criterios
Nombre de la Tarea: Modificar Criterio	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio:22/3/2013	Fecha Fin:23/3/2013
Programador Responsable: Tania Pérez	
Descripción: Se muestra una interfaz donde el usuario debe buscar el criterio por un criterio de búsqueda y así seleccionarlo a partir de un listado y modificarlo.	

Tabla 29 Descripción de la TI Modificar Criterio

Tarea de Ingeniería 6 Eliminar Criterio

TAREA DE INGENIERÍA	
Número Tarea: 6	Historia de Usuario (no.2): Administrar Criterios
Nombre de la Tarea: Eliminar Criterio	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio:22/3/2013	Fecha Fin:23/3/2013
Programador Responsable: Tania Pérez	
Descripción: Se muestra una interfaz donde el usuario debe buscar el criterio por un criterio de búsqueda y así seleccionarlo a partir de un listado y eliminarlo.	

Tabla 30 Descripción de la TI Eliminar Criterio

Tarea de Ingeniería 7 Insertar Experto

TAREA DE INGENIERÍA	
Número Tarea: 7	Historia de Usuario (no.3): Administrar Expertos
Nombre de la Tarea: Insertar Experto	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4

Fecha Inicio:24/3/2013	Fecha Fin: 25/3/2013
Programador Responsable: Tania Pérez y Malena Pereda	
Descripción: Se muestra una interfaz donde el usuario debe introducir los datos del experto.	

Tabla 31 Descripción de la TI Insertar Experto

Tarea de Ingeniería 8 Modificar Experto

TAREA DE INGENIERÍA	
Número Tarea: 8	Historia de Usuario (no.3): Administrar Expertos
Nombre de la Tarea: Modificar Experto	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio:24/3/2013	Fecha Fin: 25/3/2013
Programador Responsable: Tania Pérez y Malena Pereda	
Descripción: Se muestra una interfaz donde el usuario debe buscar el experto por un criterio de búsqueda y así seleccionarlo a partir de un listado y modificarlo.	

Tabla 32 Descripción de la TI Modificar Experto

Tarea de Ingeniería 9 Eliminar Experto

TAREA DE INGENIERÍA	
Número Tarea: 9	Historia de Usuario (no.3): Administrar Expertos
Nombre de la Tarea: Eliminar Experto	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio:24/3/2013	Fecha Fin: 25/3/2013
Programador Responsable: Tania Pérez y Malena Pereda	
Descripción: Se muestra una interfaz donde el usuario debe buscar el experto por un criterio de búsqueda y así seleccionarlo a partir de un listado y eliminarlo .	

Tabla 33 Descripción de la TI Eliminar Experto

Tarea de Ingeniería 10 Importar Fichero

TAREA DE INGENIERÍA	
Número Tarea: 10	Historia de Usuario (no.4): Manipular ficheros
Nombre de la Tarea: Importar Fichero	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.6

Fecha Inicio:26/3/2013	Fecha Fin: 27/3/2013
Programador Responsable: Tania Pérez y Malena Pereda	
Descripción: Se muestra un botón que te permita importar los ficheros que ya se hayan realizado.	

Tabla 34 Descripción de la TI Importar Fichero

Tarea de Ingeniería 11 Exportar Fichero

TAREA DE INGENIERÍA	
Número Tarea: 11	Historia de Usuario (no.4): Manipular ficheros
Nombre de la Tarea: Exportar Fichero	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.6
Fecha Inicio:28/3/2013	Fecha Fin: 29/3/2013
Programadores Responsables: Tania Pérez y Malena Pereda	
Descripción: Se muestra un botón que te permita exportar los ficheros.	

Tabla 35 Descripción de la TI Exportar Fichero

Tarea de Ingeniería 12 Exportar como PDF

TAREA DE INGENIERÍA	
Número Tarea: 12	Historia de Usuario (no.5): Exportar documento

	con información
Nombre de la Tarea: Exportar como PDF	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio:29/3/2013	Fecha Fin: 30/4/2013
Programador Responsable: Tania Pérez	
Descripción: Se muestra un botón que te permita exportar la información obtenida en un documento pdf.	

Tabla 36 Descripción de la TI Exportar como PDF

Tarea de Ingeniería 13 Técnica AHP

TAREA DE INGENIERÍA	
Número Tarea: 13	Historia de Usuario (no.6): Utilizar Técnicas AHP y Jerarquía AHP
Nombre de la Tarea: Técnica AHP	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 1
Fecha Inicio:31/3/2013	Fecha Fin: 7/4/2013
Programador Responsable: Tania Pérez	

Descripción: Se muestra un botón que te permita utilizar esta técnica y mostrar una gráfica de barra con los resultados obtenidos.

Tabla 37 Descripción de la TI Técnica AHP

Tarea de Ingeniería 14 Técnica Jerarquía AHP

TAREA DE INGENIERÍA	
Número Tarea: 14	Historia de Usuario (no.6): Utilizar Técnicas AHP y Jerarquía AHP
Nombre de la Tarea: Técnica Jerarquía AHP	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.6
Fecha Inicio:31/4/2013	Fecha Fin: 7/4/2013
Programador Responsable: Malena Pereda	
Descripción: Se muestra un botón que te permita utilizar esta técnica y mostrar una gráfica de barra con los resultados obtenidos.	

Tabla 38 Descripción de la TI Técnica Jerarquía AHP

Tarea de Ingeniería 15 Técnica Planning Game

TAREA DE INGENIERÍA	
Número Tarea: 15	Historia de Usuario (no.7): Utilizar el Planning Game
Nombre de la Tarea: Técnica Planning Game	
Tipo de Tarea: Desarrollo	Puntos estimados: 0.6

(Desarrollo / Corrección / Mejora / Otra (especificar))	
Fecha Inicio:8/4/2013	Fecha Fin: 9/4/2013
Programador Responsable: Tania Pérez	
Descripción: Se muestra un botón que te permita utilizar esta técnica y mostrar una lista con los resultados obtenidos.	

Tabla 39 Descripción de la TI Técnica Planning Game

Iteración 2

Tarea de Ingeniería 1 Técnica Estrategia Cognitiva

TAREA DE INGENIERÍA	
Número Tarea: 1	Historia de Usuario (no.8): Utilizar la Estrategia Cognitiva
Nombre de la Tarea: Técnica Estrategia Cognitiva	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio:10/4/2013	Fecha Fin: 19/4/2013
Programador Responsable: Malena Pereda	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar una lista con los resultados obtenidos.	

Tabla 40 Descripción de la TI Técnica Estrategia Cognitiva

Tarea de Ingeniería 2 Técnica Asignación Numérica

TAREA DE INGENIERÍA	
Número Tarea: 2	Historia de Usuario (no.9): Técnica asignación numérica
Nombre de la Tarea: Técnica asignación numérica	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio: 20/4/2013	Fecha Fin: 22/4/2013
Programador Responsable: Tania Pérez y Malena Pereda	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar una lista con los resultados obtenidos.	

Tabla 41 Descripción de la TI Técnica asignación numeral

Tarea de Ingeniería 3 Técnica 100 puntos

TAREA DE INGENIERÍA	
Número Tarea: 3	Historia de Usuario (no.10): Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC)
Nombre de la Tarea: Técnica 100 puntos	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4

Fecha Inicio: 23/4/2013	Fecha Fin: 24/4/2013
Programador Responsable: Malena Pereda	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar una lista con los resultados obtenidos.	

Tabla 42 Descripción de la TI Técnica 100 puntos

Tarea de Ingeniería 4 Técnica Votación jerárquica acumulativa (VHC)

TAREA DE INGENIERÍA	
Número Tarea: 4	Historia de Usuario (no.10): Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC)
Nombre de la Tarea: Técnica Votación jerárquica acumulativa (VHC)	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio: 23/4/2013	Fecha Fin: 24/4/2013
Programador Responsable: Malena Pereda	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar una lista con los resultados obtenidos.	

Tabla 43 Descripción de la TI Técnica Votación jerárquica acumulativa (VHC)

Tarea de Ingeniería 5 Utilizar el Método Unión Matriz Wiegers-VOP

TAREA DE INGENIERÍA

Número Tarea: 5	Historia de Usuario (no.11): Utilizar el Método Unión Matriz Wiegers-VOP.
Nombre de la Tarea: Utilizar el Método Unión Matriz Wiegers-VOP.	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio: 25/4/2013	Fecha Fin: 26/4/2013
Programador Responsable: Malena Pereda	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar una lista con los resultados obtenidos.	

Tabla 44 Descripción de la TI Utilizar el Método Unión Matriz Wiegers-VOP

Tarea de Ingeniería 6 Planning Combinado con el AHP

TAREA DE INGENIERÍA	
Número Tarea: 6	Historia de Usuario (no.12): Utilizar Técnica Planning Combinado con el AHP.
Nombre de la Tarea: Planning Combinado con el AHP.	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 0.4
Fecha Inicio: 27/4/2013	Fecha Fin: 28/4/2013
Programador Responsable: Tania Pérez	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar gráficas con los resultados obtenidos.	

Tabla 45 Descripción de la TI Planning Combinado con el AHP

Iteración 3

Tarea de Ingeniería 1 Burbuja

TAREA DE INGENIERÍA	
Número Tarea: 1	Historia de Usuario (no.13): Utilizar la técnica Burbuja
Nombre de la Tarea: Burbuja	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra (especificar))	Puntos estimados: 1
Fecha Inicio:28/4/2013	Fecha Fin:30/4/2013
Programador Responsable: Malena Pereda	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar una tabla con los resultados obtenidos.	

Tabla 46 Descripción de la TI Burbuja

Tarea de Ingeniería 3 Optimización Multiobjetivo

TAREA DE INGENIERÍA	
Número Tarea: 3	Historia de Usuario (no.14): Utilizar el Optimización Multiobjetivo
Nombre de la Tarea: Optimización Multiobjetivo	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Mejora / Otra	Puntos estimados: 3

(especificar))	
Fecha Inicio:1/5/2013	Fecha Fin:21/5/2013
Programador Responsable: Tania Pérez	
Descripción: Se muestra un opción en la barra de menú que permite utilizar esta técnica y mostrar una lista con los resultados obtenidos.	

Tabla 47 Descripción de la TI Optimización Multiobjetivo

Anexo 4 Pruebas de Aceptación

Prueba de Aceptación 4 HU Administrar Requisitos

Caso de Prueba de Aceptación	
Código: HU1_p4	**Historia de Usuario (No.1)**: Administrar requisitos
Nombre: Insertar un requisito repetido.	
Descripción: Se desea probar que el sistema no permite la duplicación de requisitos con el mismo id.	
Condiciones de ejecución: Deben de existir un requisito con el mismo id.	

Entrada/ Pasos de ejecución: Juego de datos con el id repetido .
Resultado esperado: Se muestra una ventana de error notificando la existencia de ese requisitos.

Tabla 48 Prueba de Aceptación HU1_p4

Prueba de Aceptación 5 HU Administrar Criterios

Caso de Prueba de Aceptación	
Código: HU2_p1	**Historia de Usuario (No.2):** Administrar criterios
Nombre: Insertar un criterio con datos incorrectos	
Descripción: Probar que no se puede insertar un criterio con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben de existir errores en los datos de los criterios.	
Entrada/ Pasos de ejecución: Criterios con casillas en blanco o datos con errores. El sistema comprueba que los datos estén correcto o completos.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria	

Tabla 49 Prueba de Aceptación HU2_p1

Prueba de Aceptación 6 HU Administrar Criterios

Caso de Prueba de Aceptación	
Código: HU2_p2	**Historia de Usuario (No.2):** Administrar criterios
Nombre: Eliminar un criterio.	
Descripción: Se desea probar que el sistema elimina satisfactoriamente un criterio.	
Condiciones de ejecución: Se accede a la opción de eliminación de un criterio.	
Entrada/ Pasos de ejecución: Se debe seleccionar el criterio que se desea eliminar y se debe verificar si se eliminó correctamente.	
Resultado esperado: Se debe actualizar el listado de criterios.	
Evaluación de la prueba: Satisfactoria	

Tabla 50 Prueba de Aceptación HU2_p2

Prueba de Aceptación 7 HU Administrar Criterios

Caso de Prueba de Aceptación	
Código: HU2_p3	**Historia de Usuario (No.1):** Administrar criterios
Nombre: Modificar datos de un criterio.	

Descripción: Se desea probar que el sistema permite cambiar los datos de un criterio.
Condiciones de ejecución: Juegos de datos correctos.
Entrada/ Pasos de ejecución: Se selecciona un criterio y se modifican sus datos
Resultado esperado: Se debe actualizar el listado de criterios.
Evaluación de la prueba: Satisfactoria

Tabla 51 Prueba de Aceptación HU2_p3

Prueba de Aceptación 8 HU Administrar Criterios

Caso de Prueba de Aceptación	
Código: HU2_p4	**Historia de Usuario (No.2):** Administrar criterios
Nombre: Insertar un criterio repetido.	
Descripción: Se desea probar que el sistema no permite la duplicación de criterio scon el mismo nombre.	
Condiciones de ejecución: Deben de existir un criterio con el mismo nombre.	

Entrada/ Pasos de ejecución: Juego de datos con el nombre repetido .
Resultado esperado: Se muestra una ventana de error notificando la existencia de ese criterios.

Tabla 52 Prueba de Aceptación HU2_p4

Prueba de Aceptación 9 HU Administrar Expertos

Caso de Prueba de Aceptación	
Código: HU3_p1	**Historia de Usuario (No.3):** Administrar expertos
Nombre: Insertar un expertocon datos incorrectos	
Descripción: Probar que no se puede insertar un experto con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben de existir errores en los datos de los expertos.	
Entrada/ Pasos de ejecución: Expertos con casillas en blanco o datos con errores. El sistema comprueba que los datos estén correcto o completos.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria	

Tabla 53 Prueba de Aceptación HU3_p1

Prueba de Aceptación 10 HU Administrar Expertos

Caso de Prueba de Aceptación	
Código: HU3_p2	**Historia de Usuario (No.3):** Administrar expertos
Nombre: Eliminar un experto.	
Descripción: Se desea probar que el sistema elimina satisfactoriamente un experto.	
Condiciones de ejecución: Se accede a la opción de eliminación de un experto.	
Entrada/ Pasos de ejecución: Se debe seleccionar el expertoque se desea eliminar y se debe verificar si se eliminó correctamente.	
Resultado esperado: Se debe actualizar el listado de expertos.	
Evaluación de la prueba: Satisfactoria	

Tabla 54 Prueba de Aceptación HU3_p2

Prueba de Aceptación 11 HU Administrar Expertos

Caso de Prueba de Aceptación	
Código: HU3_p3	**Historia de Usuario (No.3):** Administrar expertos

Nombre: Modificar datos de un experto.
Descripción: Se desea probar que el sistema permite cambiar los datos de un experto.
Condiciones de ejecución: Juegos de datos correctos.
Entrada/ Pasos de ejecución: Se selecciona un experto y se modifican sus datos
Resultado esperado: Se debe actualizar el listado de expertos.
Evaluación de la prueba: Satisfactoria

Tabla 55 Prueba de Aceptación HU3_p3

Prueba de Aceptación 12 HU Administrar Expertos

Caso de Prueba de Aceptación	
Código: HU3_p4	**Historia de Usuario (No.3):** Administrar expertos
Nombre: Insertar un experto repetido.	
Descripción: Se desea probar que el sistema no permite la duplicación de expertos con el mismo nombre.	
Condiciones de ejecución: Deben de existir un experto con el mismo nombre.	

Entrada/ Pasos de ejecución: Juego de datos con el nombre repetido .
Resultado esperado: Se muestra una ventana de error notificando la existencia de ese expertos.

Tabla 56 Prueba de Aceptación HU3_p4

Prueba de Aceptación 13 HU Manipular Ficheros

Caso de Prueba de Aceptación	
Código: HU4_p1	**Historia de Usuario (No.4):** Manipular Ficheros
Nombre: Abrir listado de requisitos.	
Descripción: Se desea probar que el sistema pueda abrir correctamente el fichero del listado de requisitos.	
Condiciones de ejecución: Se accede a la opción de abrir un fichero que contiene la información de los requisitos.	
Entrada/ Pasos de ejecución: Se debe seleccionar la opción abrir fichero y se debe verificar que el fichero abrió correctamente actualizando la lista de requisitos.	
Resultado esperado: Se debe actualizar el listado de requisitos.	

Evaluación de la prueba: Satisfactoria

Tabla 57 Prueba de Aceptación HU4_p1

Prueba de Aceptación 14 HU Manipular Ficheros

Caso de Prueba de Aceptación	
Código: HU4_p2	**Historia de Usuario (No.4):** Manipular Ficheros
Nombre: Guardar listado de requisitos.	
Descripción: Se desea probar que el sistema pueda guardar correctamente el fichero del listado de los requisitos.	
Condiciones de ejecución: Se accede a la opción de guardar un fichero que contiene la información de los requisitos.	
Entrada/ Pasos de ejecución: Se debe seleccionar la opción guardar fichero y se debe verificar que el fichero se guardó correctamente.	
Resultado esperado: Se debe guardar un fichero con la información de todos los requisitos.	
Evaluación de la prueba: Satisfactoria	

Tabla 58 Prueba de Aceptación HU4_p2

Prueba de Aceptación 15 HU Exportar documento con información

Caso de Prueba de Aceptación	
Código: HU5_p1	**Historia de Usuario (No.5):** Exportar documento con información
Nombre: Guardar información de los resultados del proceso de priorización en PDF.	
Descripción: Se desea probar que el sistema pueda exportar correctamente la información en PDF.	
Condiciones de ejecución: Se accede a la opción de exportar en PDF que contiene la información de los resultados de la priorización .	
Entrada/ Pasos de ejecución: Se debe seleccionar la opción exportar en PDF y se debe verificar que el documento se exportó correctamente.	
Resultado esperado: Se debe exportar un documento con la información la priorización.	
Evaluación de la prueba: Satisfactoria	

Tabla 59 Prueba de Aceptación HU5_p1

Prueba de Aceptación 16 HU Utilizar Técnicas AHP y Jerarquía AHP

Caso de Prueba de Aceptación	
Código: HU6_p1	**Historia de Usuario (No.6):** Utilizar Técnicas AHP y Jerarquía AHP

Nombre: Técnica AHP.
Descripción: Se desea probar que el usuario pueda insertar los valores (de 1 al 9) de las comparaciones de los requisitos atendiendo al o a los criterios que se hayan insertado, que el sistema muestre la gráfica con el resultado de los valores relativos.
Condiciones de ejecución: Se accede a la opción de calcular los valores de los requisitos.
Entrada/ Pasos de ejecución: Se debe seleccionar la opción calcular valores de los criterios en caso de que sean más de uno y después la opción calcular valores.
Resultado esperado: Se permitirá observar en una gráfica el resultado final.
Evaluación de la prueba: Satisfactoria

Tabla 60 Prueba de Aceptación HU6_p1

Prueba de Aceptación 17 HU Utilizar Técnicas AHP y Jerarquía AHP

Caso de Prueba de Aceptación	
Código: HU6_p2	**Historia de Usuario (No.6):** Utilizar Técnicas AHP y Jerarquía AHP
Nombre: Jerarquía AHP.	
Descripción: Se desea probar que el usuario pueda dividir los requisitos en jerarquías, insertar los valores (de 1 al 9) de las comparaciones de los requisitos atendiendo al o a los criterios que se hayan insertado, que el sistema muestre la	

gráfica con el resultado de los valores relativos.
Condiciones de ejecución: Se accede a la opción de calcular los valores de los requisitos.
Entrada/ Pasos de ejecución: Se debe seleccionar la opción calcular valores de los criterios en caso de que sean más de uno y después la opción calcular valores.
Resultado esperado: Se permitirá observar en una gráfica el resultado final.
Evaluación de la prueba: Satisfactoria

Tabla 61 Prueba de Aceptación HU6_p2

Prueba de Aceptación 18 HU Utilizar el Planning Game

Caso de Prueba de Aceptación	
Código: HU7_p1	**Historia de Usuario (No.7):** Utilizar el Planning Game
Nombre: Planning Game.	
Descripción: Se desea probar que el usuario pueda dividir los requisitos en las tres pilas y que a su vez pueda asignarle un ranking a cada requisito dentro una misma pila.	
Condiciones de ejecución: Se accede a la opción de planning game.	
Entrada/ Pasos de ejecución: Se debe seleccionar la opción planning game.	

Resultado esperado: Se permitirá observar una tabla con los requisitos ordenados.
Evaluación de la prueba: Satisfactoria

Tabla 62 Prueba de Aceptación HU7_p1

Iteración 2

Prueba de Aceptación 1 HU Utilizar la Estrategia Cognitiva

Caso de Prueba de Aceptación	
Código: HU8_p1	**Historia de Usuario (No.8):** Utilizar la Estrategia Cognitiva
Nombre: Estrategia Cognitiva.	
Descripción: Se desea probar que el sistema permita, insertar cantidad de participantes de forma correcta e insertar los valores de los requisitos.	
Condiciones de ejecución: Se accede a la opción Estrategia Cognitiva.	
Entrada/ Pasos de ejecución: El sistema comprueba que la cantidad de participantes sea un número y que los valores estén dentro del rango.	
Resultado esperado: Se debe mostrar un mensaje de error en caso de que la cantidad de participantes no sea un número o los valores insertados estén fuera de rango.	
Evaluación de la prueba: Satisfactoria	

Tabla 63 Prueba de Aceptación HU8_p1

Prueba de Aceptación 2 HU Utilizar Técnica asignación numérica

Caso de Prueba de Aceptación	
Código: HU9_p1	**Historia de Usuario (No.9):** Utilizar Técnica asignación numérica
Nombre: Técnica asignación numérica.	
Descripción: Probar de que se pueda dividir los requisitos en los 3 grupos que determina la técnica.	
Condiciones de ejecución: El sistema debe permitir seleccionar en la tabla el grupo al que pertenece cada requisito y actualizar la lista de los requisitos.	
Entrada/ Pasos de ejecución: Se debe seleccionar la opción Técnica Asignación Numérica .	
Resultado esperado: El sistema debe mostrar una mensaje de confirmación sobre si se ha asignado correctamente.	
Evaluación de la prueba: Satisfactoria	

Tabla 64 Prueba de Aceptación HU9_p1

Prueba de Aceptación 3 HU Utilizar técnica de los 100 puntos y Técnica Votación jerárquica

acumulativa (VHC)

Caso de Prueba de Aceptación	
Código: HU10_p1	**Historia de Usuario (No.10):** Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC)
Nombre: Insertar cantidad de expertos para la Técnica de los 100 puntos y VHC.	
Descripción: Probar que no se puede insertar la cantidad de expertos con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben de existir errores en los datos al entrar la cantidad de expertos que se deben introducir.	
Entrada/ Pasos de ejecución: Cantidad de expertos con casillas en blanco o datos con errores, se intenta que el sistema acepte la cantidad de expertos.	
Resultado esperado: El sistema debe mostrar una ventana de error.	
Evaluación de la prueba: Satisfactoria	

Tabla 65 Prueba de Aceptación HU10_p1

Prueba de Aceptación 5 HU Utilizar técnica de los 100 puntos

Caso de Prueba de Aceptación	
Código: HU10_p2	**Historia de Usuario (No.10):** Utilizar Técnica de los 100 puntos y Técnica Votación jerárquica acumulativa (VHC)
Nombre: Asignar puntos para la Técnica de los 100 puntos y VHC.	
Descripción: Probar que no se pueden asignar la cantidad de puntos a los requisitos con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben de existir errores en los datos al entrar la cantidad de puntos que se deben introducir por los expertos a los requisitos.	
Entrada/ Pasos de ejecución: Cantidad de puntos con casillas en blanco o datos con errores, se intenta que el sistema acepte la cantidad de puntos.	
Resultado esperado: El sistema debe mostrar una ventana de error.	
Evaluación de la prueba: Satisfactoria	

Tabla 66 Prueba de Aceptación HU10_p2

Prueba de Aceptación 6 HU Utilizar el Método Unión Matriz Wiegers-VOP

Caso de Prueba de Aceptación

Código: HU11_p1	**Historia de Usuario (No.11):** Utilizar el Método Unión Matriz Wiegers-VOP
Nombre: Insertar valores.	
Descripción: Se desea probar que el sistema permita insertar los valores de los requisitos dentro del rango que permite la unión de los métodos.	
Condiciones de ejecución: Se accede a la opción Wiegers-VOP.	
Entrada/ Pasos de ejecución: El sistema comprueba que los valores insertados se encuentra dentro del rango.	
Resultado esperado: Se debe mostrar un mensaje informando que los valores se han insertado correctamente.	
Evaluación de la prueba: Satisfactoria	

Tabla 67 Prueba de Aceptación HU11_p1

Prueba de Aceptación 7 HU Utilizar el Método Unión Matriz Wiegers-VOP

Caso de Prueba de Aceptación	
Código: HU11_p2	**Historia de Usuario (No.11):** Utilizar el Método Unión Matriz Wiegers-VOP
Nombre: Calcular prioridad.	

Descripción: Se desea probar que el sistema muestre el resultado final los valores calculados.
Condiciones de ejecución: Se accede a la opción calcular dentro de la opción Wiegers-VOP.
Entrada/ Pasos de ejecución: El sistema calcula la prioridad a partir de los valores insertados.
Resultado esperado: Se debe mostrar la prioridad calculada por cada requisito.
Evaluación de la prueba: Satisfactoria

Tabla 68 Prueba de Aceptación HU11_p2

Prueba de Aceptación 8 HU Utilizar Técnica Planning Combinado con el AHP

Caso de Prueba de Aceptación	
Código: HU12_p1	**Historia de Usuario (No.12):** Utilizar Técnica Planning Combinado con el AHP.
Nombre: Planning Combinado con el AHP.	
Descripción: Se desea probar que el sistema muestre el resultado final los valores calculados.	
Condiciones de ejecución: Se accede a la opción calcular dentro de la opción PG con AHP.	

Entrada/ Pasos de ejecución: El sistema calcula la prioridad a partir de los valores insertados.
Resultado esperado: Se debe mostrar la prioridad calculada por cada requisito.
Evaluación de la prueba: Satisfactoria

Tabla 69 Prueba de Aceptación HU12_p1

Iteración 3

Prueba de Aceptación 1 HU Utilizar la Técnica Burbuja

Caso de Prueba de Aceptación	
Código: HU13_p1	**Historia de Usuario (No.13):** Utilizar la Técnica Burbuja.
Nombre: Burbuja	
Descripción: Se desea probar que el sistema muestre el resultado final los valores calculados.	
Condiciones de ejecución: Se accede a la opción calcular dentro de la opción Tec. Burbuja.	
Entrada/ Pasos de ejecución: El sistema calcula la prioridad a partir de los valores insertados.	
Resultado esperado: Se debe mostrar la prioridad calculada por cada requisito.	

Evaluación de la prueba: Satisfactoria

Tabla 70 Prueba de Aceptación HU13_p1

Prueba de Aceptación 2 HU Utilizar el Proceso Optimización Multiobjetivo

Caso de Prueba de Aceptación	
Código: HU14_p1	**Historia de Usuario (No.14):** Utilizar el Proceso Optimización Multiobjetivo
Nombre: Asignar valores correctos a los requisitos.	
Descripción: Probar que se puede asignar los valores en los rangos aceptables de dependencias (0- si entre el par Ri y Rj no tienen dependencia, 1- si se escoge Ri se debe escoger Rj, 2- si se escoge Ri no se puede escoger Rj, 3- primero debe desarrollarse Ri y después Rj, 4- a la inversa), costo y valor relativo (los dos en la escala de 1 a 9 como el AHP) a los requisitos descritos por el proceso .	
Condiciones de ejecución: El usuario introducir los valores correctos.	
Entrada/ Pasos de ejecución: Se accede a seleccionar las opciones de dependencias, costo y valores relativos de cada requisito.	
Resultado esperado: El sistema debe mostrar una ventana de informando la asignación de valores satisfactoria.	

Evaluación de la prueba: Satisfactoria

Tabla 71 Prueba de Aceptación HU14_p1

Prueba de Aceptación 3 HU Utilizar el Proceso Optimización Multiobjetivo

Caso de Prueba de Aceptación	
Código: HU14_p2	**Historia de Usuario (No.14):** Utilizar el Proceso Optimización Multiobjetivo
Nombre: Asignar valores incorrectos o incompletos a los requisitos.	
Descripción: Probar que no se puede insertar datos incompletos o incorrectos.	
Condiciones de ejecución: Deben existir juegos de datos incorrectos o incompletos.	
Entrada/ Pasos de ejecución: Juego de datos con casillas en blanco y datos con errores, se intenta que el sistema los acepte.	
Resultado esperado: Se muestra una ventana de error donde se expone un mensaje con el error cometido.	
Evaluación de la prueba: Satisfactoria	

Tabla 72 Prueba de Aceptación HU14_p2

Prueba de Aceptación 4 HU Utilizar el Proceso Optimización Multiobjetivo

Caso de Prueba de Aceptación	
Código: HU14_p3	**Historia de Usuario (No.14):** Utilizar el Proceso Optimización Multiobjetivo
Nombre: Calcular el resultado final .	
Descripción: Probar que se puede calcular el resultado final aplicando el algoritmo genético.	
Condiciones de ejecución: El usuario debe seleccionar la opción aplicar el algoritmo.	
Entrada/ Pasos de ejecución: Juego de datos correctos y completos de todos los expertos y requisitos.	
Resultado esperado: Se muestra una tabla con los requisitos que deben ser desarrollados.	
Evaluación de la prueba: Satisfactoria	

Tabla 73 Prueba de Aceptación HU14_p3

Glosario de términos

A

AHP: Proceso de Análisis Jerárquico, AHP por sus siglas Analytical Hierarchy Process. Técnica de comparación entre pares de alternativas mediante una escala de 1-9.

Asignación Numérica: Técnica para la priorización de requisitos de software permitiendo dividirlos en 3 grupos en dependencia del valor de importancia.

B

BPL: Binary Priority List, lista de prioridad binaria, técnica de comparación entre pares de requisitos para asignar prioridades.
Burbuja: Técnica que se utiliza para comparar por cada par de requisitos y priorizar.

C

CV: Cumulative Voting o 100 puntos. Técnica que permite la asignación de puntos a los stakeholders para que se los otorguen a los requisitos que ellos consideran que se deben implementar. Finalmente se calcula la frecuencia relativa de cada requisito.

E

Estrategia Cognitiva: Proceso para priorizar requisitos de software teniendo en cuenta el conocimiento de los stakeholders.

J

Jerarquía AHP: Divide los requisitos en grupos jerárquicos y en cada uno de los grupos utiliza el AHP para calcular el o los valores relativos definidos.

Jerarquía CV: Divide los requisitos en grupos jerárquicos y en cada uno de los grupos utiliza el CV para calcular el o los valores relativos definidos.

M

Matriz de Wiegers: Método que se utiliza para priorizar requisitos atendiendo a 4 criterios: penalidad, riesgo, costo y valor de importancia.

P

Proceso de Priorización: Proceso de Toma de Decisiones en el que los stakeholders deciden cuáles requisitos desarrollar.

PG: Juego de Planeamiento, PG por sus siglas en inglés Planning Game, divide los requisitos en grupos de importancia. Finalmente dentro de cada grupo se realiza un ranking descendente.

Planning Game combinado con el AHP: Técnica en la que se dividen los requisitos por los grupos establecidos en el PG para posteriormente aplicar el AHP en cada grupo.

Proceso de Optimización Multiobjetivo: Proceso mediante el cual se plantea el problema de la mochila para priorizar los requisitos de software maximizando el valor de importancia, minimizando el costo de implementación atendiendo a restricciones de dependencia entre los requisitos. Finalmente se resuelve el problema a través del algoritmo genético NSGA II .

R

Requisitos de Software: Características, funcionalidades, propiedades que el sistema debe tener.

Ranking: Actividad que se realiza para ordenar aspectos según criterio(s) definido(s).

T

Toma de Decisiones: Proceso en el que se analizan un conjunto de alternativas a las que se le asignan prioridades teniendo en cuenta criterios de selección predefinidos, para a partir de esos valores decidir el conjunto de alternativas o la alternativa a desarrollar.