Universidad de las Ciencias Informáticas

Facultad 4

Selección de portafolio o cartera de proyectos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Hismel Himely Álvarez

Tutores:

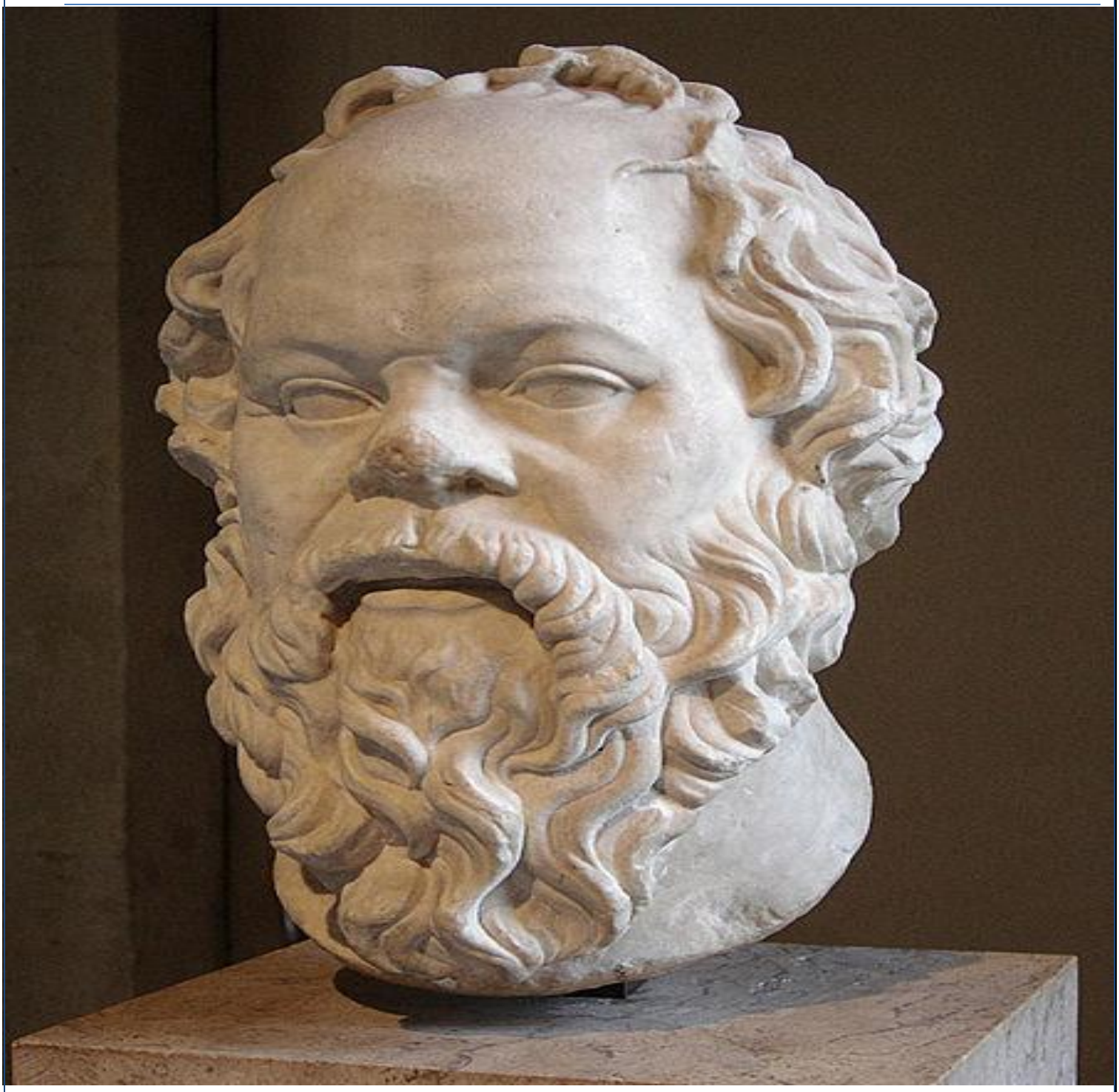
Ing. Yanko Hernández Valdés

Ing. Javier Ruiz García

La Habana, junio de 2012

"Año 53 de la Revolución"

FRASE



"Solo es útil el conocimiento que nos hace mejores"

Sócrates

DEDICATORIA

A mi familia y amistades en general, en especial a mi mamá y a mi hermana por su excepcional amor, cariño, confianza, comprensión, apoyo y amistad que han depositado en mí durante toda la vida, por contribuir tanto en mi formación, por el esfuerzo de todos estos años lo cual ha hecho posible este maravilloso momento y que además ha permitido convertirme en el hombre que soy hoy.

AGRADECIMIENTOS

A mi mamá por ser madre y padre durante estos largos 25 años, ofreciéndome su amor incondicional en todo momento, tanto en lo momentos buenos como en los momentos difíciles. Por la forma en que me crio y me educo logrando convertirme en una persona de bien. Por todos los momentos difíciles que ha tenido en la vida y a pesar de eso siempre se mantuvo fuerte y segura frente a los obstáculos que se interponían en nuestras vidas. Por su apoyo incondicional, por darme toda la fuerza y confianza que he necesitado. Este momento tan especial y maravilloso ha sido posible gracias al fruto de mi madre, por no dejarse caer frente a los obstáculos que la puesto la vida, por ser capaz de levantarse y luchar por sus hijos en lugar de rendirse y llorar. Por todos esos tristes momentos y por todo el amor depositado en sus hijos ella ha sido mi razón de ser una persona honesta, sencilla, estudiada, preocupada, preparada para la vida, ha sido mi razón de vivir y de existir, ha sido la razón de que hoy yo sea un ingeniero y todo esas cosas se lo debo a ella, motivo por el cual siempre estaré en deuda con ella y eso nunca lo podre olvidar.

A mí querida hermana por estar orgullosa de mí, por su cariño, su amor, su ternura y por todo lo que ella representa para mí. Por estar siempre presente en todos los momentos de mi vida, por todo el apoyo que me ha brindado en todos estos años, por tener tanta confianza en mí y ayudarme a salir de los momentos difíciles. A ella que ha sido como una segunda madre para mí, preocupándose por las cosas que me preocupan y me inquietan, por ser consejera y amiga. Por los momentos felices que hemos pasado juntos y también por los tristes momentos donde hemos aprendido que lo único que importa es el amor que nos tenemos.

AGRADECIMIENTOS

A "Mariscal" como cariñosamente le decimos, gracias por toda la formación que me has dado en toda mi vida. A ti que me has visto crecer guiándome como un padre gracias por todos los regaños, consejos, por la educación, por sembrar valores en mí desde niño que hoy me hacen ser el hombre que soy. Agradecerte por ser parte de todos los momentos felices de mi vida, gracias por siempre estar ahí, de corazón te digo que no tengo en la vida como pagar todo lo que has hecho por mí, espero que este sea uno de los frutos que con amor supiste sembrar.

A mi familia en general, que tanto se ha preocupado por mí.

A mis amistades del IPCVE "Vladimir Ilich Lenin" del grupo 7 por compartir 3 años de estudio, sacrificio, devoción, preocupación, amistad, momentos alegre y momentos felices; en particular al "grupito" de San Miguel, las PQT y demás; y en especial para tres personas que desde que los conocí han sido para mí los hermanos que nunca tuve: Alex, Yasniel y Yosel, a los tres muchas gracias por su amistad brindada en todo este tiempo, no existe nada para mí que se compare con su amistad, siempre voy a estar en deudas con ustedes.

A todos los profesores que he tenido durante todos mis años de estudios, los cuales han contribuido a mi formación desde el circulo infantil hasta la universidad, pero en especial a mi profesora del alma, guía, consejera y amiga de mi grupo en la vocacional: Coralia, gracias por todo el cariño y amor con el que nos educó.

AGRADECIMIENTOS

A todas las amistades que he conocido durante estos años de carrera y de las cuales nunca podré olvidar cada momento que compartimos juntos; a las pocas amistades que quedan de nuestro primer año en la Universidad que juntos compartimos ese tan difícil y duro primer año en la brigada 8107. Quiero mencionar en especial a cuatro personas que han estado presentes en todo este tiempo y que ha jugado un papel importante en mi vida en la Universidad:

Danay y Aryanna gracias por todo su carinno y apoyo brindado, no tengo como agradecerles, nunca las olvidare; a Fredy y Zamora por estar juntos desde primer año compartiendo el mismo apartamento, por estar juntos en cada fiesta fuese Amanezcos, Ranchón, Chikola, San Pedro, los Campismos, por lo momentos difíciles de estudios, por las noches sin dormir, por estar siempre a mi lado brindándome su amistad.

A las muchachitas del 141101 por soportarme en este último año conviviendo con ustedes, por ayudarme a tener un año mejor de lo que yo esperaba. Gracias a todas por compartir su tiempo conmigo, además del hecho de haber realizado nuevas amistades a las que no olvidare como Anay e Ivis. Un especial agradecimiento para "Mirita" por ser probablemente la persona que más se haya preocupado por mí en todo este curso, por ayudarme y aconsejarme en los momentos difíciles, por malcriarme, soportarme y brindarme su cariño con esa forma natural y sencilla que la caracteriza . A todas ustedes muchas gracias.

AGRADECIMIENTOS

A mis tutores por todo su apoyo y por ayudarme a salir de los momentos difíciles y enseñarme el camino correcto a seguir. Por ser tan exigentes por inculcarme el deber de hacer las cosas bien y con el nivel que se requiere.

A todas aquellas personas que de una forma u otra colaboraron para que fuera posible este momento y que me brindaron su apoyo en momentos de desaliento.

A todos mis más profundos agradecimientos.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2012

Autores:

_____

Hismel Himely Álvarez

Tutores:

_____

Ing. Yanko Hernández Valdés

_____

Ing. Javier Ruiz García

DATOS DE CONTACTO

Tutor: Yanko Hernández Valdés: Ingeniero Industrial, Instructor, graduado en el año 2005. Su dirección de correo es yanko@uci.cu.

Tutor: Javier Ruiz García: Ingeniero en Ciencias Informáticas, Instructor, recién graduado, graduado en el año 2008. Su dirección de correo es jruiz@.uci.cu.

RESUMEN

Resumen

La gestión de portafolios de proyectos es una función de administración de gran importancia en las empresas, para lograr resultados eficientes y concretar las estrategias trazadas, también para tener claridad de las prioridades de los proyectos y que estos no sufran retrasos en su ejecución. Teniendo en consideración la necesidad de profundizar y mejorar las técnicas usadas por empresas productoras de software para la selección de grupos de proyectos a desarrollar, se presenta una solución matemática informatizada que suple carencias a este campo y tiene como aporte la inclusión de criterios actuales, múltiples y personalizables para la selección, así como la posibilidad de secuenciar en el tiempo los proyectos que conforman el portafolio. Para lograr esto se desarrolló un sistema que implementa la solución matemática elaborada en el presente trabajo. Este resultado se obtiene de la fusión de diferentes materias, entre las que se encuentran los fundamentos matemáticos con las Teorías de Grafos y Combinatoria. Para el proceso de desarrollo de software se hizo uso de la metodología RUP, apoyándose en UML como lenguaje de modelado con Visual Paradigm for UML 8.0 en la Ingeniería de Software y en la programación. Como herramientas constructoras se utilizó el gestor de base de datos MySQL 4.5 y el servidor XAMPP 1.7, diseñándose el modelo relacional con la herramienta DBDesigner 4.0 y la implementación se realizó sobre la plataforma java con el Netbeans IDE 7.1. Con la utilización del sistema obtenido se mejora la selección de portafolios de proyectos informáticos, garantizando un menor tiempo de aplicación con mayor flexibilidad de configuración.

ÍNDICE

ÍNDICE

ÍNDICE

INTRODUCCIÓN

Introducción

La idea de seleccionar qué proyectos son más factibles de realizar por una empresa desarrolladora de software, siempre ha sido una decisión crucial, pues dicha empresa tiene la posibilidad de escoger qué proyectos desean realizar, además, la metodología empleada para distribuir su presupuesto y seleccionar los proyectos que deben ser ejecutados por estas empresas, ha evolucionado con el paso de los años.

En el momento de realizar la planificación la empresa debe tener en cuenta que, si la cantidad de proyectos candidatos supera el número que se puede atender, entonces es necesario seleccionar un portafolio o cartera de proyectos acorde a la capacidad de la organización. Los proyectos que conforman el portafolio se realizan en un determinado período de tiempo, facilitando la gestión efectiva del trabajo para el que están destinados a realizar y permiten el cumplimiento de los objetivos estratégicos de negocio.

Con la evolución de las organizaciones surge la idea de seleccionar y planificar portafolios de proyectos, apareciendo la necesidad de encontrar una metodología o modelo global que sea capaz de resolver la mayor cantidad de problemas de selección y planificación temporal de carteras o portafolios. Estas metodologías permitieron que las empresas comenzaran a apoyar sus decisiones de selección en soluciones matemáticas para distribuir su presupuesto. La mayoría de las soluciones matemáticas que se han propuesto incluyen una gran cantidad de cálculos matemáticos que resultan tediosos para su aplicación manual, razón por la cual estas se comenzaron a automatizar. (Carazo Fernández, et al., 2010)

Con la incorporación de la informática al proceso de selección de portafolios de proyectos se incorporaron nuevos métodos de selección en los cuales la idea es no solo seleccionar las empresas con los recursos disponibles sino, además, determinar el portafolio que maximice los resultados. Por tal razón y por el desarrollo alcanzado por las empresas en los últimos años se ha creado la alternativa de escoger las mejores combinaciones de proyectos y no proyectos individuales como se hacía en sus inicios. (Carazo Fernández, et al., 2010)

En la actualidad la selección de portafolios de proyectos ha tenido un desarrollo notable en cuanto a los métodos y técnicas empleadas en sus inicios, así como los beneficios y resultados que han permitido el uso de las nuevas metodologías aplicadas en esta esfera, pero estos métodos no son todavía los idóneos para poder aplicarlos en la Universidad de Ciencias Informáticas (UCI), debido a que la mayoría de los sistemas informáticos que se han desarrollado con el objetivo de realizar un mejor proceso de selección

de portafolios de proyectos, presentan deficiencias como son: la poca flexibilidad en cuanto al manejo de los criterios de selección y el hecho de no ser multiplataforma.

La Vicerrectoría de Producción en la UCI constituye la entidad rectora del proceso de producción de software en la institución, donde una de sus principales tareas es participar como guía metodológica en la producción de software en diversas temáticas para lo cual es necesario recopilar información que le permita realizar estas negociaciones basadas en la realidad. Esto solamente es posible con la incorporación de nuevos conocimientos que permitan una renovación permanente de los métodos, procedimientos y herramientas que se utilizan.

En estos momentos la Universidad cuenta con personal joven e inexperto en su mayoría pero que tiene la capacidad y preparación para una rápida asimilación de las tecnologías necesarias. Los recursos materiales no están en correspondencia tampoco con las necesidades actuales, pero el principal déficit está hoy en la capacidad de dirección y organización de esta producción y en la escasez de procedimientos y herramientas que permitan garantizar el éxito de los compromisos asumidos. (Tamayo, 2010)

Desde sus inicios la Universidad se ha enfrentado con una serie de obstáculos en la ejecución de los proyectos, por parte de los clientes o por deficiencias propias de la institución. Algunos de estas deficiencias son: atraso en el tiempo de ejecución de los proyectos, poca experiencia de los líderes y principales directivos de los proyectos, no disponer del personal y los recursos tecnológicos necesarios para poder realizar de manera satisfactoria los proyectos asumidos, no tener clara la envergadura del mismo en la fase inicial, los procesos no son definidos correctamente por parte de los clientes o el proyecto no ofrece las ganancias esperadas. (Tamayo, 2010)

“La evaluación de proyectos ofrece significativas ventajas al sistema nacional de ciencia e innovación tecnológica, pues se ha convertido en un elemento fundamental para su organización, priorizar los que presentan mayores posibilidades de éxito, reducir el tiempo entre la obtención de los resultados y su introducción en la práctica social, impulsar el desarrollo de los programas priorizados, favorecer el ambiente creativo de los investigadores, garantizar un mejor empleo de los recursos y estimular los mejores esfuerzos y resultados.” (Tamayo, 2010)

El proceso de aceptación y evaluación de proyectos es realizado por el Grupo de Inteligencia Corporativa,

la Alternativa Bolivariana para la Exportación de Tecnologías (ALBET) y la Vicerrectoría de Producción, donde el método empleado no es lo suficientemente eficiente, pues no resulta preciso en el momento de escoger entre varias propuestas de proyectos.

De esta forma no existe actualmente un método formal para establecer una comparación entre los proyectos que se desean desarrollar, por lo que estos se ejecutan en la medida en que se adquiere el compromiso y se asignan a la entidad responsable dentro de la UCI. La falta de un método de evaluación coherente para la toma de decisiones, hace más difícil la selección de los proyectos que deben ser ejecutados, sobre todo cuando no se puede determinar cuál de todos los proyectos aceptados es el más conveniente, y ni la lógica ni la intuición son de ayuda en estos casos.

Por tal razón es necesario encontrar un camino para determinar qué proyecto resulta más favorable realizar que otro, tanto a corto como a largo plazo, debido a que si no se pueden desarrollar todas las propuestas a la vez, si sería necesario y conveniente utilizar los recursos disponibles en los proyectos adecuados.

Con el fin de satisfacer la excesiva demanda de software de diversas esferas que llegan a la Universidad, es necesario desarrollar un método que permita evaluar a los proyectos de manera efectiva, haciendo uso de técnicas de evaluación que permita valorarlos y establecer un orden de prioridad con un mínimo de esfuerzo. En el año 2009 surge la idea de desarrollar por estudiantes y profesores de la UCI la herramienta MySPP, esta aplicación tiene como objetivo realizar un proceso selección de portafolios de proyectos de manera satisfactoria y acorde a las necesidades.

La herramienta MySPP se basa en un modelo matemático multivariado en forma de árbol donde se miden de igual manera todos los proyectos potenciales, sus variables y ponderaciones se adicionan, modifican o eliminan haciendo de este un sistema flexible y adaptable a los cambios crecientes del entorno. Aunque la herramienta esta accesible y operable aún necesita mejoras en el modelo matemático que permitan la inclusión de una secuenciación de los proyectos en el tiempo y el cambio de modelo multivariado en forma de árbol por los grafos, así como la disminución del déficit de memoria por el aumento significativo de las combinaciones de los portafolios de proyectos a evaluar.

Analizando lo antes expuesto se plantea el siguiente **problema a resolver**: ¿Cómo garantizar la inclusión de una secuenciación de los proyectos en el tiempo de la herramienta MySPP, que permita superar el

déficit de memoria del ordenador, por el aumento significativo de las combinaciones a evaluar?

Para darle solución al problema se define el siguiente **objetivo general:** Definir un Modelo Matemático que sustente la herramienta MySPP y sus funcionalidades en el sistema y que posibilite la secuenciación en el tiempo de los proyectos que conforman el portafolio.

El problema planteado se enmarca en el **objeto de estudio:** Los procesos de gestión de proyectos de software.

En la investigación se conciben como **campo de acción** los modelos matemáticos.

Se plantea como **idea a defender**, que la implementación de un modelo matemático para la herramienta MySPP permitirá espaciar en el tiempo los portafolios de proyectos, obteniendo así una planificación en el tiempo de ejecución de los mismos.

Para alcanzar el objetivo general de la investigación se debe dar cumplimiento a los siguientes **objetivos específicos:**

- ✓ Analizar, diseñar e implementar en la herramienta los cambios introducidos al modelo matemático.
- ✓ Secuenciar en función del tiempo los proyectos que conforman el portafolio, partiendo de ajustes y aportes al modelo matemático.
- ✓ Realizar pruebas a la herramienta que demuestren la disponibilidad operativa del software.
- ✓ Elaborar la documentación pertinente al proceso de obtención del sistema para la selección de portafolios de proyectos.
- ✓ Confeccionar un manual de usuario que brinde información detallada sobre como trabajar con la herramienta a desarrollar.

Tareas de investigación:

- ✓ Estudiar los aspectos teóricos que sustentan el proceso de selección de proyectos informáticos a desarrollar.
- ✓ Estudiar el modelo matemático que sustenta a MySPP.
- ✓ Consulta de bibliografía referente a los principales aspectos teóricos, actividades y procesos que se manejan en la selección de proyectos informáticos para su realización.
- ✓ Consulta de bibliografía referente al modelo matemático que sustenta MySPP.
- ✓ Secuenciación en función del tiempo de los proyectos que conforman el portafolio, partiendo de

ajustes y aportes al modelo matemático.

- ✓ Análisis, diseño e implementación de las mejoras y ajustes en el modelo matemático al sistema para la selección de portafolios de proyectos.
- ✓ Confección del Manual del usuario del funcionamiento del sistema para la selección de portafolios de proyectos.
- ✓ Realización de pruebas a la interfaz del sistema.

Métodos de Investigación:

Teóricos:

- ✓ Analítico-Sintético: Luego del estudio realizado de la presente investigación se decide utilizar este método ya que el mismo permitió hacer un análisis de las teorías, documentos y bibliografías relacionadas con los temas de selección de portafolios de proyectos, modelos matemáticos, teoría de grafos y combinatorias entre otros. Además se empleó para realizar un análisis y estudio de las normas que plantean cómo seleccionar un portafolio de proyectos, con el fin de tener una idea más clara de la situación a resolver, así como la extracción de los elementos más importantes que se relacionan con el objeto de estudio de la investigación.
- ✓ Análisis Histórico-Lógico: A través de la investigación realizada se decidió hacer uso de este método, pues el mismo permitió realizar una investigación de las diferentes metodologías existentes para seleccionar un portafolio de proyectos. También posibilitó analizar la evolución cualitativa y cuantitativa; así como el desarrollo de los métodos y técnicas de selección de portafolios de proyectos hasta la actualidad.
- ✓ Modelado: Durante el análisis de la presente investigación se aprobó emplear este método, para representar a través del diseño planteado, una posible interfaz de la aplicación para modelar el sistema.

Estructura capitular

El trabajo está estructurado en cuatro capítulos tal y como se describe a continuación:

CAPÍTULO I FUNDAMENTACIÓN TEÓRICA. En este capítulo se expone la fundamentación teórica del tema. Incluye una descripción detallada del objeto de estudio, con la finalidad de comprender el proceso de selección de portafolios de proyectos. Además se hace un resumen de las metodologías, los lenguajes,

las herramientas, las tecnologías, las arquitecturas y los patrones utilizados en el proceso de desarrollo del software.

CAPÍTULO II MODELO MATEMÁTICO.

En este capítulo se explica el modelo matemático para la selección de portafolios de proyecto. Además, se muestra un ejemplo de selección de carteras de proyectos.

CAPÍTULO III CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA. En este capítulo se describe el negocio, se enumeran los requisitos del sistema para tener un mejor entendimiento de lo que se quiere concebir. Se definen los actores del sistema, se describen los casos de uso y se presenta el diagrama de casos de uso. Además, se realiza el análisis y el diseño del sistema.

CAPÍTULO IV IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA. En este capítulo se presentan los diagramas de despliegue y componentes como objetivo primordial de la fase de implementación. Conjuntamente se realizan las pruebas del sistema a las cuáles se sometió la aplicación.

Capítulo I Fundamentación teórica

Introducción

En el desarrollo del presente capítulo se realizará un estudio del estado del arte de la gestión de portafolios de proyectos, además se analizaran los principales conceptos que tienen relación con la selección y planificación temporal de carteras de proyectos. Se muestra una representación más específica del objeto de estudio, para que exista una mejor comprensión de cómo se ejecuta el proceso de selección de carteras de proyectos, por una empresa u organización desarrolladora de software. También se presentan herramientas similares que permiten la selección de portafolios de proyectos, con el objetivo de profundizar más en el tema que se investiga. Por último se exponen las metodologías, lenguajes, herramientas, tecnologías, arquitecturas y patrones que serán empleados durante el desarrollo de la investigación y que permitirán la implementación del sistema que se desea.

1.1 Conceptos fundamentales

A partir del análisis de la presente investigación y del estudio de las diversas bibliografías, relacionadas con el proceso de selección de portafolios de proyectos, se observa que la mayor parte de los autores definen en sus trabajos los términos proyecto y portafolio o carteras de proyecto, debido a que consideran estos conceptos vitales para las investigaciones relacionadas con la selección de portafolios. Al finalizar el estudio de las bibliografías consultadas coincidimos con las definiciones de los autores Ana Fernández Carazo, Rafael Caballero Fernández, Trinidad Gómez Núñez entre otros, dado a que las definiciones que ellos dan en el artículo "Evaluación y clasificación de las técnicas utilizadas por las organizaciones, en las últimas décadas, para seleccionar proyectos" son las que coinciden con el tema de la investigación.

Estos autores definen que: "*Un proyecto es un esfuerzo temporal, único e irrepetible que, consumiendo un conjunto de recursos, busca satisfacer unos objetivos específicos en un período de tiempo determinado*." (Carazo Fernández, et al., 2010), lo cual se aplica a la investigación. Además, dichos autores definen que: "*Un portafolio o cartera de proyectos es un conjunto de proyectos que, llevados a cabo en un determinado período de tiempo, comparten una serie de recursos y entre los que pueden existir relaciones de complementariedad, incompatibilidad y sinergias producidas por compartir costes y beneficios derivados de la realización de más de un proyecto a la vez."* (Carazo Fernández, et al., 2010)

Aparte de las reiteradas ocasiones en que los autores, que tratan el tema de selección de portafolios de proyectos, mencionan los términos proyecto y portafolio o carteras de proyectos, se apreció durante el estudio realizado que también hacen énfasis en el concepto de modelo matemático. Esto se debe a la idea de aprobar las decisiones de selección de proyectos basados en soluciones matemáticas, esto se traduce en desarrollar sistemas basados en modelos matemáticos que permitan la selección de proyectos. Por tal razón se realiza un estudio del estado del arte, donde se analizan las definiciones dadas de los modelos matemáticos. Una de las definiciones analizadas es la que da el Ing. Rafael Cuevas Mijangos, el cual plantea que: *"Un modelo es producto de una abstracción de un sistema real eliminando las complejidades y haciendo suposiciones pertinentes, se aplica una técnica matemática y se obtiene una representación simbólica del mismo."* (Mijango, 2010)

Por otra parte encontramos la definición dada por los autores Juan Alberto Rodríguez Velázquez y Cristina Steegmann Pascual, los cuales plantean que*: "Un modelo matemático es una descripción, en lenguaje matemático, de un objeto que existe en un universo no-matemático."* (Rodríguez Velázquez, et al., 2010)

De manera general la mayoría de las definiciones dadas coinciden con la que plantea el profesor Juan Sánchez Ramos donde él define a los modelos matemáticos como entidades del sistema donde los atributos se representan mediante variables matemáticas y las actividades se describen mediante funciones matemáticas que interrelacionan las variables. (Ramos, 2001-2002)

Además de investigar sobre los modelos matemáticos, se realizó una búsqueda de la definición del término secuenciación dado el interés que se tiene de poder realizar una distribución de los portafolios en el tiempo. De manera general por secuencia/secuenciación se entiende como la ordenación de una serie de elementos, uno detrás de otro, de tal manera que tal ordenación implica una cierta relación entre las partes ordenadas. (Sánchez, 2010) Esta definición es la que se decidió aprobar debido a que la misma mantiene la idea acerca de la distribución de los proyectos en el tiempo, lo cual no es más que una secuencia de los proyectos realizados. Existen otras definiciones, pero no coincidimos con ellas debido a que no nos resultan factibles para el procedimiento que se quiere realizar.

1.2 Sistemas similares

A continuación se muestran las herramientas que se analizaron durante el estudio del proceso de selección de portafolios de proyectos.

1.2.1 Sistemas Internacionales

Project Selector 1.3 (Ver Anexo 1 Herramientas Similares)

El Project Selector Proptima calcula una cartera de proyectos o cartera de inversión óptima cuando hay un gran número de oportunidades de negocios, cada oportunidad tiene sus costos y beneficios, y la libertad de elección está limitada por los recursos limitados. Una inversión o proyecto puede tener requisitos y condiciones que deben cumplirse, lo que significa que los proyectos pueden ser dependientes entre sí. Es compatible para Windows.

Esta herramienta selecciona un portafolio de proyecto óptimo entre un número de proyectos candidatos pero no satisface nuestros requisitos a cumplir debido a que solo selecciona un portafolio de proyecto, no posee flexibilidad en el manejo de los criterios y es compatible solo para el sistema operativo Windows.

Portfolio Optimization 1.0 (Ver Anexo 1 Herramientas Similares)

El Modelo de Optimización de Cartera, calcula los coeficientes de capital óptimos para una cesta de inversiones financieras que garantiza el máximo rendimiento al mínimo de riesgos. El diseño único del modelo le permite ser aplicado a cualquier instrumento financiero o las carteras de negocios. La capacidad de aplicar el análisis de optimización de una cartera de negocios representa un marco excelente para la conducción de la asignación de capital, y las inversiones. (Finanzas Personales : Gestión de la cartera : cartera de Optimización de 1, 2002)

Las características principales del modelo de optimización de cartera son: La facilidad y la flexibilidad de entrada, con la ayuda de mensajes incrustados; capacidad de especificar el número de unidades presentes en cada producto o negocio y las limitaciones de la cartera optimizada; también presenta la opción para garantizar que el regreso no se deteriore a expensas de riesgo, así como la pantalla gráfica intuitiva con el resultado de la simulación, incluyendo el análisis de probabilidad de determinado nivel de retorno de destino. (Finanzas Personales : Gestión de la cartera : cartera de Optimización de 1, 2002)

Esta herramienta calcula los coeficientes de capital óptimos para una cesta de inversiones financieras que garantiza el máximo rendimiento al mínimo de riesgos pero no cumple a cabalidad nuestros requisitos actuales pues no permite que el usuario cree sus propios criterios de selección, no es multiplataforma pues es compatible con los sistemas operativos Windows y Macintosh y es privativo.

1.2.2 Sistema nacional

Herramienta MySPP (Ver Anexo 1 Herramientas Similares)

La herramienta MySPP fue desarrollada por estudiantes y profesores de la Universidad de las Ciencias Informáticas UCI en el año 2009, con el objetivo de realizar un proceso de selección de portafolios de proyectos de manera satisfactoria y acorde a las necesidades. Esta herramienta esta implementada a partir de un modelo matemático que tiene una estructura en forma de árbol, donde se miden de igual manera todos los proyectos potenciales, permite gestionar sus variables y ponderaciones, haciendo de este un sistema flexible y adaptable a los cambios crecientes del entorno. Está desarrollada en el lenguaje java, permitiendo portabilidad y uso en las diferentes plataformas.

La herramienta permite la selección de portafolios de proyectos que maximizan los resultados de la institución, pero no satisface todas las necesidades actuales como la secuenciación de los proyectos en el tiempo, ni el déficit de memoria de los ordenadores al realizar las combinaciones de los portafolios, por lo que será usada como fuente reutilizable en cuanto a ideas y funcionalidades.

Al concluir el análisis sobre el papel que ha jugado la informática en el desarrollo de aplicaciones que permitan la selección de portafolios o carteras de proyectos, se pudo apreciar de manera general que, tanto en el ámbito nacional como internacional ninguno de los sistemas presentados realiza el proceso de selección de portafolios de proyectos, de manera que cumpla con los requisitos que se desean. Estos requisitos son: el desarrollo de una herramienta multiplataforma que realice la selección de proyectos de manera tal que exista una flexibilidad en cuanto a la creación y estructuración de los criterios y que permita la distribución de los proyectos en el tiempo. El sistema Portfolio Optimization 1.0 se asemeja bastante a lo que se desea realizar, pero no cumple la condición de ser multiplataforma, además de no tener una gran flexibilidad en cuanto al manejo de los criterios de selección por parte de los usuarios. Por su parte la herramienta MySPP brinda la posibilidad de ser multiplataforma, pero esta herramienta no permite realizar la distribución de los portafolios, necesitando ajustes en el modelo matemático en que se basa.

Es válido decir que en el año 2010 se realizó una tesis de maestría en la UCI cuyo autor es la Ing. Karina Sánchez Tamayo, donde se plantea un método efectivo para evaluar proyectos informáticos, así como establecer un orden de prioridad que ayude a la toma de decisiones. Para el desarrollo de la presente investigación se tuvieron en cuenta muchos elementos tratados por la Ing. Karina Sánchez, sobre todo en la lógica de negocio de la gestión de proyectos en la UCI. Al final de la tesis la autora concluye

recomendado que se realice una herramienta que agilice el proceso de selección de carteras de proyectos. Después de realizar un estudio de las principales herramientas y bibliografías, que brindan y proponen como ejecutar un proceso de selección de portafolios de manera satisfactoria, se decide desarrollar una herramienta de que sea capaz de realizar dicho proceso, acorde a la necesidades que tenga cada empresa.

1.3 Proceso de desarrollo de software

Un proceso de software detallado y completo suele denominarse Metodología, define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Las metodologías se desarrollan con el objetivo de dar solución a los problemas existentes en la producción de software, que cada vez son más complejos. Estas engloban procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software. No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable. (Jacobson, et al., 2000)

La comparación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. A grandes rasgos, si se toma como criterio su filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, se nombran Metodologías Tradicionales.

Otras metodologías, denominadas Metodologías Ágiles, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

Entre las Metodologías Ágiles y Tradicionales más reconocidas se encuentran XP (del inglés Extreme Programming) y RUP (del inglés Rational Unified Process) respectivamente.

XP

XP está concebida para dirigir las necesidades específicas del desarrollo de software conducido por equipos pequeños. Esta metodología es más adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Está centrada en potenciar la comunicación desarrollador-cliente desde el primer día. Es considerada ligera, flexible, predecible, de bajo riesgo, y no por ello menos científica. Entre otras ventajas pueden

mencionarse los pocos requerimientos de documentación y planificación, así como la exigencia de tener siempre el cliente disponible para el desarrollo, implicando una mejor correspondencia entre el producto y la necesidad del negocio. (Beck, 2000)

RUP

RUP es uno de los procesos más generales, estructurado y adaptable a las características y necesidades de cualquier proyecto. Propone 6 disciplinas ingenieriles (Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Despliegue) y 3 de apoyo (Gestión de Configuración y Cambios, Gestión de Proyecto, Ambiente) que tienen lugar en algunas o todas las fases que define (Inicio, Elaboración, Construcción y Transición).

Este proceso unificado tiene tres características distintivas: (ECURed, 2012)

- ✓ Dirigido por Casos de Uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo.
- ✓ Centrado en la Arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- ✓ Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos.

RUP permite trabajar con precisión y calidad, perfeccionado el software en cualquier momento del desarrollo, logrando así, un sistema con la robustez necesaria, independientemente del tiempo disponible. Si se maneja bien el conjunto de procesos, es posible realizar todos los artefactos que propone RUP independientemente del tamaño del equipo de desarrollo.

Con el uso de esta metodología, es posible encomendar tareas específicas en otras personas involucradas indirectamente en el proyecto; sin embargo, con XP se hace necesario que los colaboradores

estén involucrados completamente en el desarrollo del producto. Por otra parte, la documentación que se obtiene mediante RUP contribuye a lograr un mejor entendimiento del sistema por parte del equipo de desarrollo, y sirve de referencia para posteriores trabajos; en cambio XP posee pocos requerimientos de documentación y planificación. Por estas razones, se elige como metodología de desarrollo de software para la solución que se propone, la cual utiliza UML (del inglés Unified Modeling Language) para preparar todos los esquemas de un sistema de software.

UML

UML es un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Con este lenguaje es posible diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas.

UML incluye conceptos semánticos, notación y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como, generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (Jacobson, et al., 2000)

1.4 Herramientas CASE para el modelado

La mayoría de las herramientas CASE (del inglés Computer Aided Software Engineering) hacen uso de UML. Estas herramientas se consideran aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software de forma tal que se reduzca el costo de las mismas en términos de tiempo y dinero. Entre las más utilizadas a nivel mundial se encuentran Visual Paradigm y Rational Rose Enterprise Edition.

Rational Rose Enterprise Edition

Rational Rose Enterprise es el producto más completo de la familia Rational Rose, es una herramienta CASE propietaria, desarrollada por Rational Corporation, basada en el Lenguaje Unificado de Modelado. Permite crear los diagramas que se van generando durante el proceso de desarrollo del software y posee un gran número de estereotipos predefinidos que facilitan el proceso de modelación.

Rational Rose Enterprise Edition posee características adicionales como: (GSInnova, 2010)

- ✓ Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
- ✓ Soporte Enterprise Java Beans 2.0.
- ✓ Capacidad de análisis de calidad de código.
- ✓ Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- ✓ Integración con otras herramientas de desarrollo de Rational.
- ✓ Capacidad para integrarse con cualquier sistema de control de versiones.

Visual Paradigm 5.0

Visual Paradigm [1]es una herramienta CASE propietaria con licencia gratuita, que soporta el ciclo de vida completo del desarrollo de software. Visual Paradigm ofrece: (Sierra, 2008)

- ✓ Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Disponibilidad en múltiples plataformas.

El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad y con menor costo. Además, permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación.

Aunque Rational Rose Enterprise Edition es una de las herramientas más potentes para el modelado visual, y una de las más utilizadas en el área de la informática, es un software propietario y no es multiplataforma, inconvenientes que limitan su uso. Considerando estos aspectos, se selecciona Visual Paradigm como herramienta para el modelado de la solución que se propone, siendo esta una herramienta multiplataforma que provee fácil integración con el resto de las herramientas de desarrollo.

1.5 Lenguajes y Herramientas de Programación

Java

Java es una plataforma virtual de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales (diferentes plataformas).

La plataforma Java consta de las siguientes partes:

- ✓ El lenguaje de programación mismo.
- ✓ La máquina virtual de Java, que permite la portabilidad en ejecución.

[1] Disponible en: http://www.visual-paradigm.com/download/

- ✓ El API [2](del inglés Application Programming Interface) de Java, una biblioteca estándar para el lenguaje.

El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smalltalk que a éste. Incorpora sincronización y manejo de tareas en el lenguaje mismo e incorpora interfaces como un mecanismo alternativo a la herencia múltiple de C++.

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado *bytecode* [3] que luego son interpretados por una máquina virtual de java (JVM). Esta última sirve como una plataforma de abstracción entre la máquina y el lenguaje permitiendo que se pueda escribir el programa una vez, y correrlo en cualquier lado. También existen compiladores nativos de Java, tanto software libre como no libre.

Se decide optar por el lenguaje Java debido a que se encuentra bajo la licencia GNU GPL **(**GNU General Public License**)** lo que la convierte en una plataforma libre. Además, brinda una excelente portabilidad y compatibilidad entre los distintos sistemas operativos, lo que hace a la herramienta completamente operacional sobre distintas plataformas. Esto último es unas de las cosas fundamentales y la razón por la cual se decide escoger a Java como lenguaje de desarrollo, pues hasta el momento se han desarrollado sistemas que permiten la selección de portafolios de proyectos, pero solo en plataformas como Windows y Macintosh, y con la ayuda de Java podemos lograr lo que se desea: la realización de un sistema que brinde la selección de portafolios de proyectos que sea multiplataforma.

Netbeans 7.1

NetBeans [4]es un entorno de desarrollo integrado es, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas con facilidad. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación como Ruby, C/C++ o PHP.

[2] Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como capa de abstracción

[3] Código intermedio más abstracto que el código máquina

[4] Disponible en: http://netbeans-ide.softonic.com/descargar

Otro aspecto importante es que Netbeans es multiplataforma, permitiendo así, que funcione en diversos sistemas operativos como Windows, Mac, Linux o Solaris. Con NetBeans es posible elaborar potentes aplicaciones de escritorio, para la Web y para dispositivos portátiles (móviles o Pocket PC). La programación mediante este se realiza a través de componentes de software modulares, también llamados módulos, que le aportan gran funcionalidad y versatilidad. (NetBeans, 2011)

Eclipse

Eclipse [5]es una comunidad de código abierto cuyos proyectos se centran en la construcción de una plataforma de desarrollo abierta compuesta por marcos extensibles, herramientas y tiempos de ejecución para la construcción, implementación y administración de software a través del ciclo de vida. Un gran y vibrante ecosistema de importantes proveedores de tecnología, nuevas empresas innovadoras, universidades, instituciones de investigación y particulares amplía, complementan y apoyan la plataforma Eclipse. (Eclipse, 2012)

La plataforma NetBeans y Eclipse RCP son más similares que diferentes. Ambos proporcionan un marco para los desarrolladores de aplicaciones de escritorio. En ambos casos, un gran número de funciones se proporcionan fuera de la caja, de un marco de acoplamiento, a un sistema de acción, para actualizar las instalaciones, y mucho más; asi como un rico conjunto de API (del inglés Application Programming Interface) se proporciona, junto con muchos tutoriales y preguntas frecuentes, y varios libros. Por lo que las diferencias existentes entre ellos no son notables en el sentido de decidir que entorno de desarrollo emplear. No obstante en el caso de la herramienta que se desea desarrollar en la presente investigación, se decide optar por Netbeans debido a que Eclipse fue liberado bajo la Common Public License y luego fue re-licenciado bajo la Free Software Foundation, las cuales son incompatibles con la Licencia General de GNU (GNU GPL), licencia que usa Java como lenguaje de programación y el propio Netbeans.

[5] http://www.eclipse.org/

DB Designer 4.0

DBDesigner [6]es un sistema de base de datos de diseño visual de base de datos que integra diseño, modelado, creación y mantenimiento en un único entorno sin fisuras. Combina características profesionales y una interfaz de usuario clara y sencilla de ofrecer la forma más eficiente para gestionar sus bases de datos. (fabFORCE.net, 2011)

DBDesigner es desarrollado y optimizado para el código abierto de MySQL-la base de datos para apoyar a los usuarios de MySQL con una potente herramienta de diseño y gratuita. (fabFORCE.net, 2011)

DBDesigner permite construir su base de datos en un entorno intuitivo y fácil de usar, donde se tiene una representación visual de las tablas y las relaciones contenidas en el proyecto. Puede ver rápidamente los campos de una tabla o cómo cada tabla se relaciona con los demás. Después que haya finalizado, DBDesigner puede exportar el esquema de la base de datos en una secuencia de comandos sql, o simplemente conectarse directamente a una base de datos de backend y construir allí. También puede importar bases de datos ya existentes de secuencias de comandos SQL o backends[7]. Por supuesto, puede guardar el proyecto en su formato original (XML) para que toda la información se mantenga. Debido a su arquitectura de plugin, DBDesigner es fácilmente extensible para trabajar con muchos servidores de bases de datos. Por defecto viene con 2 plugins: uno para PostgreSQL y el otro para MySQL. (DBDesigner, 2011)

Se decide emplear DB Designer como herramienta para el modelado de la base de datos debido a que, esta nos permite utilizar diversas características con un diseño muy fácil de usar, además de ser una herramienta efectiva para trabajar con las bases de datos. Esto es muy factible para el sistema que se desea crear, pues con el uso de esta herramienta podemos lograr una forma efectiva de gestionar nuestra base datos, con un diseño claro y fácil de entender. Asimismo esta herramienta es muy útil para la construcción de base de datos que son muy complejas.

1.6 Patrones de arquitectura

Patrón Modelo Vista Controlador (MVC)

El patrón MVC es un patrón arquitectónico de 3 capas conceptuales definido para sistemas usuario-

[6] Disponible en http://dbdesigner.softonic.com/

[7] Término que se refiere al fin de un proceso

máquina, aplicado en la actualidad a los SID Web. Este patrón no define exactamente las 3 capas clásicas de la arquitectura (Presentación, Lógica de negocios y Datos); en su lugar define las responsabilidades y las dependencias, dependiendo de los objetivos que representa en 3 paradigmas (Modelo, Vista, Controlador). (slideshare, 2011)

A continuación se mencionan las características de cada capa: (slideshare, 2011)

Modelo: Representa a toda la información con la que opera la aplicación. Gestiona el comportamiento y los datos del dominio. Responde a las peticiones de información sobre el estado, que vienen de la Vista. Responde a instrucciones de cambio de estado, provenientes del Controlador.

Vista: Gestiona la presentación de la información de la aplicación. Todo lo relativo a la interfaz de usuario, los datos de que dispone para seguir interactuando con la aplicación. Desde la interfaz gráfica a los estímulos que recibe del usuario, visual, auditiva o sensitivamente.

Controlador: Respuesta a eventos invocados desde la Vista. Llama a la lógica de negocio para procesar y producir una respuesta. Interpreta las entradas del usuario, informando al modelo y/o a la vista de los cambios que supongan esas entradas.

Flujo de trabajo del MVC

El flujo de trabajo se basa en que el usuario realiza una acción que generará un evento, luego el Controlador recibe el evento y lo traduce en una petición al Modelo (aunque también puede llamar directamente a la Vista). En caso de ser necesario el modelo llama a la Vista para su actualización, donde la Vista puede solicitar datos al Modelo para realizar la actualización. Por último el Controlador recibe el control.

Beneficios de uso del MVC:

Se han ido mencionando a lo largo del epígrafe las ventajas de este patrón, sin embargo se considera importante agruparlas en este punto:

- ✓ Se consigue múltiples vistas del modelo.
- ✓ Todas las vistas están sincronizadas.
- ✓ No acoplamiento, y facilidad de evolución, para cambiar las vistas y los controladores.
- ✓ La aplicación puede soportar un tipo de interfaz para cada usuario (rol).

Inconvenientes del MVC:

- ✓ La complejidad va creciendo más rápido que el tamaño de la aplicación.
- ✓ Problemas al conectar las distintas capas.
- ✓ Acceso ineficiente, pues es necesario varias llamadas al modelo para actualizar los datos.
- ✓ La vista y el controlador son específicos para una plataforma.

Arquitectura en capas

La arquitectura en capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan. Cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa, las capas inferiores no tienen dependencias de las capas superiores, además la comunicación entre capas está basada en una abstracción que proporciona un bajo acoplamiento entre capas. (Ecured, 2012)

Algunas de las características de este tipo de arquitectura son: (Ecured, 2012)

- ✓ Descomposición de los servicios de forma que la mayoría de interacciones ocurre solo entre capas vecinas.
- ✓ Las capas de una aplicación pueden residir en la misma máquina o estar distribuidos entre varios equipos.
- ✓ Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidos.
- ✓ Cada nivel agrega las responsabilidades y abstracciones del nivel inferior.
- ✓ Muestra una vista completa del modelo y a la vez proporciona suficientes detalles para entender las relaciones entre capas.
- ✓ No realiza ninguna suposición sobre los tipos de datos, métodos, propiedades y sus implementaciones.
- ✓ Separa de forma clara la funcionalidad de cada capa.

Las definiciones de capas son:

- ✓ Capa de presentación: Esta es la que el usuario puede ver en su ordenador, es donde se tratan los datos que se van a mostrar. Se intenta que en esta capa haya el mínimo de procesamiento. Esta capa se comunicará solamente con la capa de negocio.
- ✓ Capa de negocios: En esta capa esta la lógica, se reciben las peticiones del usuario, y tras ejecutar una acción se le envía las respuestas del proceso. Esta capa se comunica, como se ha dicho, con la de presentación, la cual le envía peticiones y esta le responde con los resultados. Y también se comunica con la capa de datos para pedirle datos.
- ✓ Capa de datos: Es donde se accede a los datos. Se hace referencia a uno o más gestores de BD que realizan el almacenamiento, modificación y consulta de los datos. Recibe peticiones desde la capa de negocios.

Beneficios de la arquitectura en capas:

- ✓ Facilita que se pueda descomponer la aplicación en varios niveles de abstracción.
- ✓ Facilita la evolución del sistema, ya que los cambios solo deben de afectar a la capa donde se encuentre la modificación.
- ✓ Si la interfaz accede a la misma función, no se repetirá código. Lo que conlleva la ventaja de una mayor facilidad de mantenimiento de la aplicación, entre otros.
- ✓ Si se añade un nuevo formulario no ocasionará verdaderos quebraderos de cabeza, ya que los formularios ya no dependen unos de otros, con lo que el cambiar el flujo de trabajo es algo trivial.
- ✓ Los formularios ya no acceden de forma independiente a los datos, ya que no se accede a través de ellos, al modificar los datos, esto implica que no se nos mostrará diferencia alguna.

Inconvenientes de la arquitectura en capas:

- ✓ Todos los sistemas no podrán ser estructurados en capas.
- ✓ Y aun pudiendo ser estructurado en capas, la separación entre una y otra no es trivial. Ya no sólo porque para un desarrollador no lo es, sino también porque muchos lenguajes y/o frameworks no están preparados para ello.

Para el sistema que se quiere desarrollar, las desventajas que presenta esta arquitectura son irrelevantes ya que se desea un sistema de fácil estructuración y será desarrollado con tecnología que facilite el trabajo con dicha arquitectura. El Modelo Vista Controlador presenta demasiados inconvenientes como

son: la complejidad va creciendo más rápido que el tamaño de la aplicación, pueden existir problemas al conectar las distintas capas, así como acceso ineficiente, ya que es necesario realizar varias llamadas al modelo para actualizar los datos. Además la vista y el controlador son específicos para una plataforma.

1.7 Sistemas Gestores de Bases de Datos y Servidores

Un Sistema Gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Por tanto debe permitir: (Collector, 2011)

- ✓ Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- ✓ Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD
- ✓ Manipular la base de datos: realizar consultas, actualizarla, generar informes.

Algunas de las características deseables en un Sistema Gestor de base de datos SGBD son: (Collector, 2011)

- ✓ Control de la redundancia: La redundancia de datos tiene varios efectos negativos (duplicar el trabajo al actualizar, desperdicia espacio en disco, puede provocar inconsistencia de datos) aunque a veces es deseable por cuestiones de rendimiento.
- ✓ Restricción de los accesos no autorizados: cada usuario ha de tener unos permisos de acceso y autorización.
- ✓ Cumplimiento de las restricciones de integridad: el SGBD ha de ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad.

Por tanto el Sistema Gestor de Bases de Datos que deseamos utilizar debe permitir en primer lugar definir una base de datos, donde se pueda especificar tipos, estructuras y restricciones de datos. En segundo lugar debe permitir construir la base de datos con el fin de guardar los datos. Y por último poder manipular la base de datos con el objetivo de realizar consultas, actualizarlas y generar informes.

MySQL 4.5

MySQL [8]es un sistema de gestión de bases de datos relacional, licenciado bajo la Licencia Pública

[8] Disponible en http://dev.mysql.com/downloads/

General de GNU (del inglés GNU GPL). Su diseño multi-hilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright [9]del código fuente del servidor SQL, así como también de la marca. (MySQL, 2011)

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario. (MySQL, 2011)

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Las principales características de este gestor de bases de datos son las siguientes: (MySQL, 2011)

- ✓ Aprovecha la potencia de sistemas multiprocesador gracias a su implementación multi-hilo.
- ✓ Soporta gran cantidad de tipos de datos para las columnas.
- ✓ Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, entre otros.).
- ✓ Gran portabilidad entre sistemas.
- ✓ Soporta hasta 32 índices por tabla.
- ✓ Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

PostgreSQL

PostgreSQL [10]es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. (PostGreSQL-es, 2010)

[9] Derecho que tiene un autor, sobre todas y cada una de sus obras y que le permite decidir en qué condiciones han ser éstas reproducidas y distribuidas

[10] Disponible en http://www.postgresql.org/download/

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilo para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (PostGreSQL-es, 2010)

A continuación se enumeran las principales características de este gestor de bases de datos: (PostGreSQL, 2011)

Implementación del estándar SQL92/SQL99.

- ✓ Soporta distintos tipos de datos, además del soporte para los tipos base, soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, entre otros. También permite la creación de tipos propios.
- ✓ Incorpora una estructura de datos array.
- ✓ Incorpora funciones de diversa índole: Manejo de fechas, geométricas, orientadas a operaciones con redes, entre otras.
- ✓ Permite la declaración de funciones propias, así como la definición de disparadores.
- ✓ Soporta el uso de índices, reglas y vistas.
- ✓ Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

La razón por la cual se decide emplear MySQL como sistema gestor de base de datos es porque esta herramienta tiene características, las cuales fueron enunciadas en los epígrafes anteriores, que se asemejan a la mayoría de los requisitos y elementos que se quieren para el software. Una de las ventajas del uso de MySQL es la implementación multi-hilo que brinda y mantiene un buen nivel de seguridad. Además es válido decir que MySQL es uno de los gestores más usados en el mundo(1) y esto viene dado por su facilidad de configuración e instalación, así como la eficiencia, velocidad y bajo consumo que presenta, tres características importantes que debe cumplir MySPP para que pueda ser ejecutado en un ordenador con escasos recursos sin ningún problema.

Xampp 1.7

Xampp [11]es un servidor independiente de la plataforma en la cual se esté ejecutando, es un software totalmente libre y se encuentra bajo la licencia GNU GPL. Básicamente Xampp consiste en una recopilación de aplicaciones y servidores donde podemos encontrar al servidor Web Apache, el motor de Bases de Datos MySQL, los lenguajes de programación PHP, Perl y otros servicios más. El nombre Xampp proviene del acrónimo X (para cualquiera de los Sistema Operativos en la que se está ejecutando), Apache, MySQL, PHP, Perl. (AplicacionesEmpresariales.com, 2008)

Xampp se encuentra disponible para los sistemas Microsoft Windows, GNU/Linux, Solaris y MacOSX. Permite montar todo un ambiente de servicios pre-configurado y listos para hacer uso del mismo, despreocupándose de conectar algunos componentes, configurarlo y realizar conexiones. El proyecto Xampp es absolutamente software libre, esto quiere decir que todas las librerías, módulos, lenguajes de programación y servidores son considerados como Software Libre.

Xampp es la construcción de una versión fácil de instalar para los desarrolladores que desean entrar en el mundo web y de los servidores Apache. Como ventajas a los desarrolladores que no tienen muchos conocimientos de configuraciones, Xampp nos provee con una configuración totalmente funcional, activando todas las funciones y conexiones. Un punto a tener en cuenta, y el proyecto Xampp en su sitio oficial lo recalca, es que la configuración por defecto puede ser un punto positivo desde la perspectiva del usuario poco experimentado, pero no es buena desde el punto de vista de la seguridad. (AplicacionesEmpresariales.com, 2008)

Esto indica que no es lo suficientemente seguro para aquellos sistemas de producción o ambientes grandes, lo cual no es un inconveniente para la aplicación que se quiere desarrollar, dada la sencillez de la herramienta comparada con los grandes sistemas o ambientes de producción. El uso de Xampp facilita todo el trabajo relacionado con los servidores de bases de datos, esto se debe a su fácil configuración e instalación, así como su principal característica de ser un software libre.

Conclusiones

- ✓ El estado del arte realizado permitió demostrar la necesidad de desarrollar en la UCI, un sistema para la selección de portafolios de proyectos al tener en cuenta que, del análisis de las soluciones

[11] Disponible en: http://www.apachefriends.org/en/xampp-windows.html

similares existentes tanto en el mundo como en Cuba, ninguna se ajusta a las necesidades y exigencias actuales.

- ✓ Se realizó un estudio de las tecnologías, metodologías y herramientas necesarias a la investigación, con el fin de seleccionar las que serán utilizadas para el desarrollo de la aplicación. Fundamentando la selección y asegurando con su uso, la correcta construcción de un sistema de selección de portafolios de proyectos, implementada a través de una solución matemática basada en criterios actuales, múltiples y personalizables.

- ✓ Según el estudio realizado en esta investigación, se concreta que el perfil tecnológico para la realización de la solución que se propone queda compuesto por Java como lenguaje de programación y Netbeans como entorno de desarrollo. La metodología de desarrollo a utilizar es RUP, que se adecua a las características del proyecto y apoyándose en UML como lenguaje de modelado. Como herramientas de modelado se empleó Visual Paradigm como herramienta CASE, asi como DBDesigner como herramienta para el modelado de la base de datos. Finalmente se optó por escoger a MySQL como sistema gestor de base de datos y XAMPP como servidor.

Capítulo II Modelo Matemático

Introducción

En este capítulo se explica el funcionamiento del modelo matemático para la selección de portafolios de proyecto. Se explica como se estructuran y clasifican los criterios de selección, además se aborda sobre como se realiza el cálculo de la importancia de los proyectos, así como el esfuerzo o capacidad productiva de la empresa. También se muestra como se producen las combinaciones entre los portafolios para luego pasar a la selección de las carteras de proyectos. Al final del capítulo se muestra un ejemplo de selección de portafolios de proyectos para que sea más fácil el entendimiento del mismo.

2.1 Clasificación de los criterios de selección

Los criterios se clasifican en nodo inicial, interno y nodo final. En el caso de los nodos finales se clasifican en cuantitativos o cualitativos y en positivo o negativo.

- ✓ Nodo Inicial: Es el primer criterio que se inserta en el grafo de criterios, no contiene ponderación alguna ni se le asigna valor.
- ✓ Interno: Criterio cuyo valor es calculado a partir del impacto de los nodos que se derivan de él.
- ✓ Nodo Final: Criterio del cual no se deriva ningún nodo y su valor es insertado directamente por el usuario.

Un criterio nodo final se clasifica en positivo cuando el número asignado a este en cada proyecto es directamente proporcional al impacto que implica este valor en el proyecto; y se clasifica en negativo cuando el impacto en la importancia del proyecto es maximizada a medida que sea menor el valor asignado a este.

Los nodos finales, cuyos valores son numerables, se clasifican en cuantitativos, mientras que los nodos finales con valores no numerables para las cuales se establecen desgloses o categorías ponderadas ejemplo: Bueno, Regular y Malo con valores de 90, 50 y 15 respectivamente, se clasifican en cualitativos.

2.2 Estructura de los criterios de selección

Intuitivamente podemos decir que un grafo es un conjunto de puntos que pueden estar conectados entre sí dos a dos mediante aristas. Los puntos representarían los elementos del problema y las aristas

representarían la existencia o no de la relación.

Un grafo simple G (V, E) consta de V, un conjunto no vacío de vértices, y de E, un conjunto de pares no ordenados de elementos distintos de V. A esos pares se les llama aristas o lados. (Chacón, 2011)

Un grafo dirigido o dígrafo G = (V, E) consta de un conjunto V de vértices, un conjunto E de aristas, que son pares ordenados de elementos de V. (Chacón, 2011).

Los criterios de selección están estructurados en un grafo dirigido y el nodo inicial es el primer criterio que se inserta, el cual no toma valor en ningún momento de la selección y su objetivo es el de agrupar a los criterios del primer nivel. Los criterios que se derivan del criterio nodo inicial son los internos, a los cuales tampoco se les asigna ningún valor ya que estos van a depender del peso que se les asigne a los criterios nodos finales. Los criterios nodos finales son aquellos que se ramifican a partir de los criterios internos y es en ellos donde el usuario introduce un valor determinado. A partir del peso asignado a los criterios nodos finales se tendrá el valor de la importancia de cada nodo final, lo que dará paso al calculo de la importancia de los criterios internos, para finalmente obtener la importancia del criterio nodo inicial, lo cual no es mas que la importancia del tiene el proyecto. Todos los criterios tienen una ponderación que es completamente configurable, pero debe cumplir que la suma de las ponderaciones de los criterios que se derivan de un mismo nodo (criterio) sea 1. El número de criterios y su organización es completamente variable evidenciando flexibilidad en el modelo.

2.3 Cálculo de la Importancia de un Proyecto

La importancia de un proyecto (*I*) es la relevancia del mismo dependiendo de los criterios de selección especificados. (Carazo Fernández, et al., 2010) Cada proyecto tiene un grafo de valores asociados, el cual es estructuralmente igual al de los criterios. Para calcular la I_r del proyecto r-ésimo se deben llenar todos los valores referentes a los criterios hojas.

Los criterios internos se calculan a partir de la siguiente fórmula:

$$X_{f,d} = \sum_{i=1}^{n} \left(P_{d,i} * X_{d,i} \right)$$

Dónde: $d, f \in N;\ P_{d,i}, X_{d,i}, X_{f,d} \in \mathrm{R};\ 0 \leq P_{d,i} \leq 1;\ 0 \geq \sum_{i=1}^{n} P_{d,i};\ 0 \leq X_{d,i} \leq 1;\ 0 \leq X_{f,d} \leq 1$.

D: Indicador del criterio del cual $X_{f,d}$ es el valor *i-ésimo* en el proyecto que se analiza. En $P_{d,i}$ y $X_{d,i}$ se referencia a que estos son los valores *i-ésimos* de los hijos del criterio $X_{f,d}$.

$X_{f,d}$: Valor del criterio *d-ésimo* nodo del criterio *f-ésimo*.

I: Indicador de criterio.

N: Cantidad de criterios.

$X_{d,i}$: Valor asociado al criterio *i-ésimo* nodo del criterio *d-ésimo*.

$P_{d,i}$: Ponderación del criterio asociado al valor$X_{d,i}$.

F: Indicador del nodo inicial del criterio.

La importancia es un caso particular del cálculo de los criterios internos y su diferencia reside en que sólo se recorren los criterios que se derivan del nodo inicial.

$$I_r = \sum_{i=1}^{b} (P_{1,i} * X_{1,i})$$

Dónde: $X_{1,i}$, I_r,$P_{1,i} \in R$, $0 \leq P_{1,i} \leq 1$, $0 \geq \sum_{i=1}^{n} P_{1,i}$, $0 \leq X_{1,i} \leq 1$.

I: Importancia o relevancia de un proyecto dependiendo de los criterios de selección dados.

R: Indicador de proyecto.

$X_{1,i}$: Valor del proyecto en el criterio *i-ésimo* derivado del nodo inicial.

$P_{1,i}$: Ponderación *i-ésima* del criterio *i-ésimo* derivado del nodo inicial.

B: grado del criterio nodo inicial.

El valor de $X_{f,i}$ se calcula dependiendo de la clasificación del criterio que se analice.

Nodo Final Positivo:

$$X_{f,d} = {V_{r,f,d}}\Big/{\max_{s}(V_{t,f,d})}$$

Dónde: *s*, *t* ∈ N; $V_{r,f,d}$, $V_{t,f,d} \in R$, $0 \leq V_{r,f,d} \leq 1$; $0 \leq V_{t,f,d} \leq 1$.

$\max_s(V_{t,f,d})$: Máximo valor que toma $V_{t,f,d}$ *d-ésimo* en los *s-ésimos* proyectos.

T: Indicador del proyecto en el max().

$V_{t,f,d}$: Valor del criterio *d-ésimo* derivado del *f-ésimo* en el proyecto *t-ésimo*.

$V_{t,f,d}$: Valor del criterio *d-ésimo* derivado del *f-ésimo* en el proyecto *r-ésimo* que refiere al proyecto actual.

S: Cantidad de proyectos.

Nodo Final Negativa:

$$X_{f,d} = {}^{\min_s(V_{t,f,d})}\Big/_{V_{r,f,d}}$$

Dónde:

$\min_s(V_{t,f,d})$: Mínimo valor que toma $V_{t,f,d}$ *d-ésimo* en los *s-ésimos* proyectos.

Como se aprecia todos los valores de los nodos finales son comparados con sus homólogos en otros proyectos logrando que desde la base se comience a establecer prioridades a los proyectos con mejor impacto en los distintos criterios.

2.4 Cálculo del esfuerzo

En este método es necesario calcular el esfuerzo disponible o capacidad productiva de la empresa y el esfuerzo necesario para completar un proyecto. (Carazo Fernández, et al., 2010)

El esfuerzo disponible de la empresa se mide en (horas/hombre) y se calcula de la siguiente manera:

$$ET = D * H * M$$

Dónde: $E \in N$, $D \in N$, $H \in N$, $M \in N$.

ET: Esfuerzo total de la empresa.
D: Días de trabajo hábiles.
H: Horas de trabajo por días.
M: Número de personas de la empresa.

El esfuerzo necesario para realizar un proyecto se mide en (horas/hombre). Se estima por el usuario empleando alguna de las técnicas que propone la disciplina Ingeniería del Software y se denota por E_r.

2.5 Selección de los portafolios de proyectos

Luego de haber calculado la importancia y esfuerzo necesario para la realización de cada proyecto se pasa a la selección de los portafolios de proyectos, con este método se puede visualizar la cantidad deseada de portafolios de proyectos posibles a desarrollar.

Las combinaciones sin repeticiones de *s* elementos disponibles, tomados en grupos de *v* elementos se calcula según la fórmula:

$$C_s^v = \binom{s}{v} = \frac{s!}{v!\,(s-v)!}$$

Dónde: s, v ϵ N, $v \leq s$.

Debido a que la extensión de los portafolios puede ser desde 1 hasta *s*, la cantidad de combinaciones posibles se calcula según la fórmula:

$$C\binom{s}{1} + C\binom{s}{2} + C\binom{s}{3} \ldots . + C\binom{s}{s}$$

Teniendo en cuenta que estos valores coinciden con los de la fila *s-ésima* del triángulo de Pascal, eliminando la primera columna del mismo, dado que esta es referente a la combinación vacía y no se tiene en cuenta para el cálculo, la suma de una fila del triángulo de Pascal se puede calcular por la fórmula:

2^s

Dónde: *s* ϵ N.

S: número de proyectos.

$2^0 = 1$

$2^1 = 1+1 = 2$

$2^2 = 1+2+1 = 4$

$2^3 = 1+3+3+1 = 8$

$2^4 = 1+4+6+4+1 = 16$

$$\binom{0}{0}$$

$$\binom{1}{0} \quad \binom{1}{1}$$

$$\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2}$$

$$\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3}$$

$$\binom{4}{0} \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4}$$

Por lo que el número máximo de portafolios de proyecto que se pueden obtener es:

$$C\binom{s}{1} + C\binom{s}{2} + C\binom{s}{3} \ldots . + C\binom{s}{s} = 2^s - 1$$

Luego de los $2^s - 1$ portafolios posibles, según la teoría combinatoria, se dejan de generar las combinaciones que sumado el esfuerzo de realización de sus proyectos excedan el esfuerzo disponible de la empresa en la etapa seleccionada, también se puede agregar como variable restrictiva el presupuesto necesario y el disponible. Estas combinaciones se van guardando en una lista con longitud igual a la cantidad de portafolios que se desea, y están ordenadas de mayor a menor según la suma de la importancia de los proyectos del portafolio.

En el caso de que existan proyectos con selección obligatoria se asegura de que estos proyectos nunca falten en los portafolios de la lista. A medida que se van generando nuevas combinaciones se comprueba que cumplan con los requisitos necesarios y se van adicionando a la lista si tienen una importancia total mayor a la de alguno de los portafolios existentes en ella y se elimina el portafolio de menor importancia en caso de que la lista supere el número deseado.

2.6 Aplicación del modelo

Para la aplicación del modelo se muestra un ejemplo simple donde se listan los 3 mejores portafolios para una empresa *"Y",* con 10 trabajadores que laboran 8 horas diarias en un espacio de tiempo de 100 días, con 5 proyectos hipotéticos y el grafo de criterios. En este ejemplo no se tiene en cuenta la restricción de presupuesto. En la siguiente tabla se reflejan los valores de los criterios nodos finales para cada proyecto, en el caso del criterio *"tipo de soporte"* que es un nodo final cualitativa se usan las categorías muy frecuente, periódico y esporádico con pesos de 150, 100 y 50 respectivamente.

Tabla 1: Valores de criterios asociados a los proyectos.

Proyecto	Tiempo de vida del soporte(meses)	Tipo de soporte	Costo inicial (dinero)	Recursos humanos	Entrada (dinero)
PRCV	30	Muy frecuente(150)	4000	30	120 000
COFER	10	Esporádico(50)	500	20	150 000
RITME	20	Periódico(100)	2000	15	20 000
ALISOFT	15	Periódico(100)	1000	25	50 000
CMV	10	Esporádico(50)	200	20	10 000

Tabla 2: Valores para el cálculo del esfuerzo.

Proyecto	Horas de trabajo estimadas para completar un proyecto(E_r)
PRCV	600
COFER	3000
RITME	2000
ALISOFT	1000
CMV	2600

El esfuerzo total de la empresa para el período dado es de:

$$ET = D * H * M$$

ET=100*8*10=8000

A continuación se calculan las importancias y el esfuerzo necesario de cada proyecto *r*.

Sustituyendo para el proyecto PRCV.

$$I_1 = P_{0,1} * \left(P_{1,3} * \frac{V_{1,1,3}}{\max\limits_r V_{r,1,3}} + P_{1,4} * \frac{V_{1,1,4}}{\max\limits_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min\limits_r V_{r,5,7}}{V_{1,5,7}} + P_{5,8} * \frac{\min\limits_r V_{r,5,8}}{V_{1,5,8}} \right) + P_{2,6} * \frac{V_{1,2,6}}{\max\limits_r V_{r,2,6}})$$

I_1=0.2*(0.6*30/30 + 0.4*150/150)+0.8*(0.15*(0.5*200/400 + 0.5*15/30) + 0.85*120000/150000)

Resolviendo I_1=0.804

$$E_1 = 600$$

Sustituyendo para el proyecto COFER.

$$I_2 = P_{0,1} * \left(P_{1,3} * \frac{V_{2,1,3}}{\max\limits_r V_{r,1,3}} + P_{1,4} * \frac{V_{2,1,4}}{\max\limits_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min\limits_r V_{r,5,7}}{V_{2,5,7}} + P_{5,8} * \frac{\min\limits_r V_{r,5,8}}{V_{2,5,8}} \right) + P_{2,6} * \frac{V_{2,2,6}}{\max\limits_r V_{r,2,6}})$$

I_2=0.2*(0.6*10/30 + 0.4*50/150) +0.8*(0.15*(0.5*200/500 + 0.5*15/20) + 0.85*150000/150000)

Resolviendo I_2= 0.8156666666666666666666666666666667

$$E_2 = 3000$$

Sustituyendo para el proyecto RITME.

$$I_3 = P_{0,1} * \left(P_{1,3} * \frac{V_{3,1,3}}{\max\limits_r V_{r,1,3}} + P_{1,4} * \frac{V_{3,1,4}}{\max\limits_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min\limits_r V_{r,5,7}}{V_{3,5,7}} + P_{5,8} * \frac{\min\limits_r V_{r,5,8}}{V_{3,5,8}} \right) + P_{2,6} * \frac{V_{3,2,6}}{\max\limits_r V_{r,2,6}})$$

I_3=0.2*(0.6*20/30 + 0.4*100/150)+0.8*(0.15*(0.5*200/2000 + 0.5*15/15) + 0.85*20000/150000)

Resolviendo I_3= 0.29

$E_3 = 2000$

Sustituyendo para el proyecto ALISOFT.

$$I_4 = P_{0,1} * \left(P_{1,3} * \frac{V_{4,1,3}}{\max\limits_r V_{r,1,3}} + P_{1,4} * \frac{V_{4,1,4}}{\max\limits_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min\limits_r V_{r,5,7}}{V_{4,5,7}} + P_{5,8} * \frac{\min\limits_r V_{r,5,8}}{V_{4,5,8}} \right) + P_{2,6} * \frac{V_{4,2,6}}{\max\limits_r V_{r,2,6}})$$

I_4=0.2 * (0.6 * 15 / 30 + 0.4 * 100 / 150) + 0.8 * (0.15 * (0.5 * 200 / 1000 + 0.5 * 15 / 25) + 0.85 * 50000 / 150000)

Resolviendo I_4= 0.388

$E_4 = 1000$

Sustituyendo para el proyecto CMV.

$$I_5 = P_{0,1} * \left(P_{1,3} * \frac{V_{5,1,3}}{\max\limits_r V_{r,1,3}} + P_{1,4} * \frac{V_{5,1,4}}{\max\limits_r V_{r,1,4}} \right) + P_{0,2} * (P_{2,5} * \left(P_{5,7} * \frac{\min\limits_r V_{r,5,7}}{V_{5,5,7}} + P_{5,8} * \frac{\min\limits_r V_{r,5,8}}{V_{5,5,8}} \right) + P_{2,6} * \frac{V_{5,2,6}}{\max\limits_r V_{r,2,6}})$$

I_5=0.2*(0.6*10/30 + 0.4*50/150) +0.8*(0.15*(0.5*200/200 + 0.5*15/20) + 0.85*10000/150000)

Resolviendo I_5= 0.217

$E_5 = 2600$

Tabla 3: Importancias y esfuerzos por proyectos.

Proyectos	Importancia	Esfuerzo
PRCV	0.804	600
COFER	0.81567	3000
RITME	0.29	2000
ALISOFT	0.388	1000
CMV	0.217	2600

Teniendo en cuenta que la empresa contiene un esfuerzo disponible de 8000 horas/hombres en el período dado y el esfuerzo necesario para realizar todos los proyectos es de 9200 horas/hombres, se evidencia la insuficiencia para cumplir con la demanda de la empresa y se concluye la necesidad de escoger los mejores proyectos para su desarrollo.

Tras la generación y organización de las combinaciones se obtienen las 3 propuestas de portafolios.

Portafolio No.1

PRCV importancia 0.804
COFER importancia 0.8156666666666668
RITME importancia 0.29000000000000004
ALISOFT importancia 0.38800000000000007

Portafolio No.2

PRCV importancia 0.804
COFER importancia 0.8156666666666668
ALISOFT importancia 0.38800000000000007
CMV importancia 0.21700000000000003

Portafolio No.3

PRCV importancia 0.804
COFER importancia 0.8156666666666668

ALISOFT importancia 0.38800000000000007

Conclusiones

- ✓ En este capítulo se realizó una explicación del modelo matemático elaborado lo cual permite un mejor entendimiento del mismo.
- ✓ Se presentó la clasificación y estructura de los criterios de selección para explicar la organización de dichos criterios.
- ✓ Se mostró el cálculo de la importancia de los proyectos y la selección de los portafolios de proyectos lo que muestra teóricamente cómo calcular la importancia de un proyecto y cómo se seleccionan los portafolios paso a paso.

  Finalmente, se expuso un ejemplo simple en el que se muestra la aplicación del modelo matemático.

Capítulo III Características, Análisis y Diseño del Sistema

Introducción

En el presente capítulo se hace una descripción del negocio mediante el diagrama de dominio, se enumeran los requisitos funcionales y no funcionales que el sistema que se propone debe poseer, permitiendo hacer una concepción general del mismo e identificar mediante diagramas de casos de usos las relaciones de los actores que se involucran con el sistema. Además, se realiza el análisis y el diseño del sistema donde se modelan los principales casos de uso seleccionados para la próxima iteración del producto, se definen los diagramas de clases del análisis de la aplicación y se especifica qué clases del análisis forman parte del caso de uso, las relaciones entre ellas y las entidades. En esta etapa se obtienen además los diagramas de secuencia. Posteriormente, mediante los diagramas del diseño, se lleva a cabo todo el flujo de los procesos, además del diagrama de clases persistentes y el modelo de datos.

3.1 Modelo del dominio

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, modelos del objeto de dominio y modelos de objetos de análisis. (Larman, 2001)

Utilizando la notación UML, un modelo del dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Pueden mostrar: (Larman, 2001)

- ✓ Objetos del dominio o clases conceptuales.
- ✓ Asociaciones entre las clases conceptuales.
- ✓ Atributos de las clases conceptuales.

En el entorno que está enmarcado el problema identificado, no se perciben claramente los procesos del negocio, ya que está altamente centrado en tecnologías informáticas, haciéndose difícil determinar el conjunto estructurado de las actividades que se desarrollan en el negocio. Por estas razones se ha

determinado realizar un Modelo Conceptual o Modelo de Dominio, teniendo en cuenta conocimientos de expertos y análisis de soluciones similares.

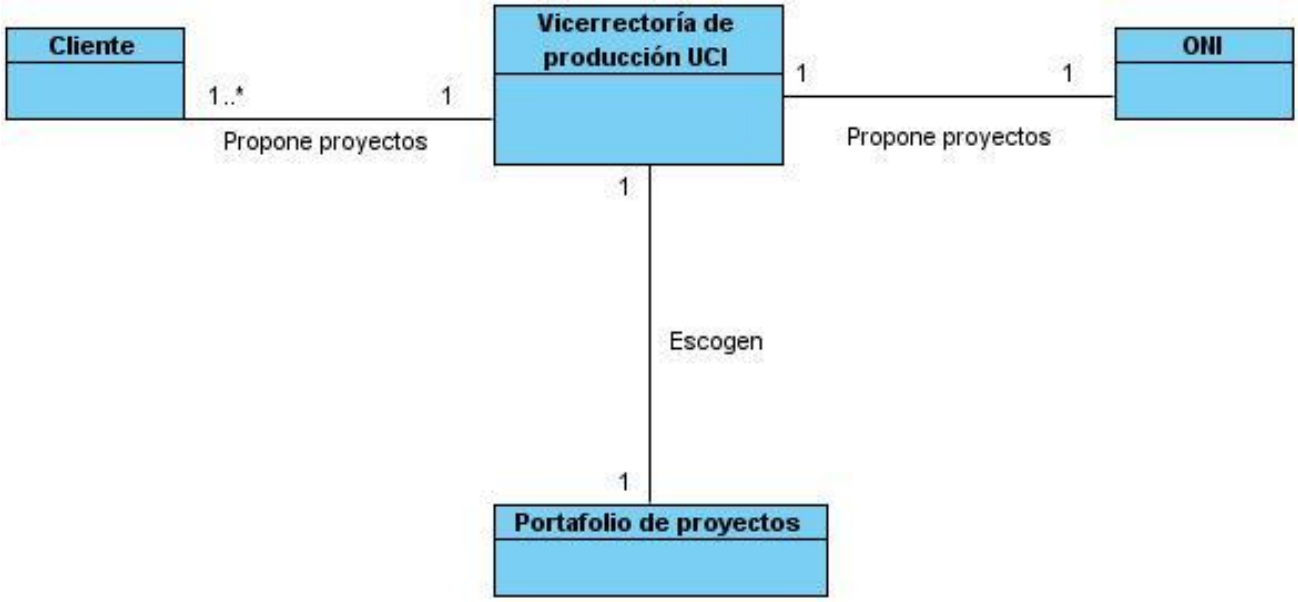


Figura 1: Diagrama del modelo del dominio

3.2 Requerimientos de la aplicación

Tras analizar el dominio del problema, es necesario definir qué es lo que debe hacer el sistema; para ello, deben ser analizadas todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan sobre lo que debe hacer el sistema como candidatas a requisitos. Los requisitos no son más que las condiciones o capacidades que tienen que ser alcanzadas por un sistema para satisfacer las necesidades del cliente. Los mismos se clasifican en requisitos funcionales [12]y requisitos no funcionales[13]. (Booch, et al., 2000)

12
Capacidades o condiciones que el sistema debe cumplir.

13
Propiedades o cualidades que el producto debe tener.

3.2.1 Requisitos funcionales

Para el sistema propuesto se determinaron los siguientes requisitos funcionales.

RF1- Gestionar criterio.

1.1 Adicionar criterio.

1.2 Eliminar criterio.

1.3 Editar criterio.

RF2- Asignar valores de criterios.

RF3- Generar portafolios.

RF4- Importar criterio.

RF5- Generar reporte.

RF6- Mostrar grafo de criterios.

RF7-Generar espaciación de los portafolios.

RF8- Conectar base de datos.

RF9- Gestionar base de datos.

9.1- Crear base de datos.

9.2- Leer base de datos existente.

RF10- Gestionar Etapa

10.1- Adicionar etapa.

10.2- Editar etapa.

10.3- Eliminar etapa.

3.2.2 Requisitos no funcionales

Ellos, para su desarrollo y análisis correcto se pueden estructurar en:

- ✓ RNF1-Requerimientos de Software los cuales para la presente investigación son: el uso de la Máquina Virtual de Java y MySQL versión 5, así como el empleo de los sistemas operativos Windows y Linux.

- ✓ RNF2-Restricciones en el diseño los cuales son: que el sistema debe contar con la característica de ser multiplataforma, además dicho sistema será desarrollado con el lenguaje de programación Java, teniendo a MySQL como sistema gestor de base de datos.
- ✓ RNF3- Requerimientos de apariencias o interfaz interna del producto entre los cuales apreciamos durante el análisis de la investigación que: el sistema debe presentar una interfaz agradable, así como permitir un fácil manejo, para que pueda ser usado por cualquier persona que posea conocimientos básicos de computación.
- ✓ RNF4- Requerimientos de seguridad los cuales según el estudio y análisis realizado son, el desarrollo de un sistema que cumpla con los principios básicos de la seguridad informática: Confiabilidad (la información o los activos informáticos solo son accedidos por las personas autorizadas), Integridad (la información o los activos informáticos solo son modificados por las personas autorizadas) y Disponibilidad (los activos informáticos o la información solo son accedidos por las personas autorizadas en el momento requerido).
- ✓ RNF5-Requerimientos de Hardware que para la investigación son el empleo de un procesador Pentium 4 con velocidad de 2 GHz o superior y el uso de una memoria RAM de 256 Mb como mínimo.

3.3 Modelo de Casos de Uso del Sistema

El Modelo de Casos de Uso ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios. Cada tipo de usuario se representa mediante un actor. Los actores utilizan el sistema al interactuar con los casos de uso (CU). Todos los actores y casos de uso del sistema forman un Modelo de Casos de Uso.

Un diagrama de casos de uso describe parte del Modelo de Casos de Uso y muestra un conjunto de casos de uso y actores con una asociación entre cada par actor/caso de uso que interactúan. (Jacobson, et al., 2000)

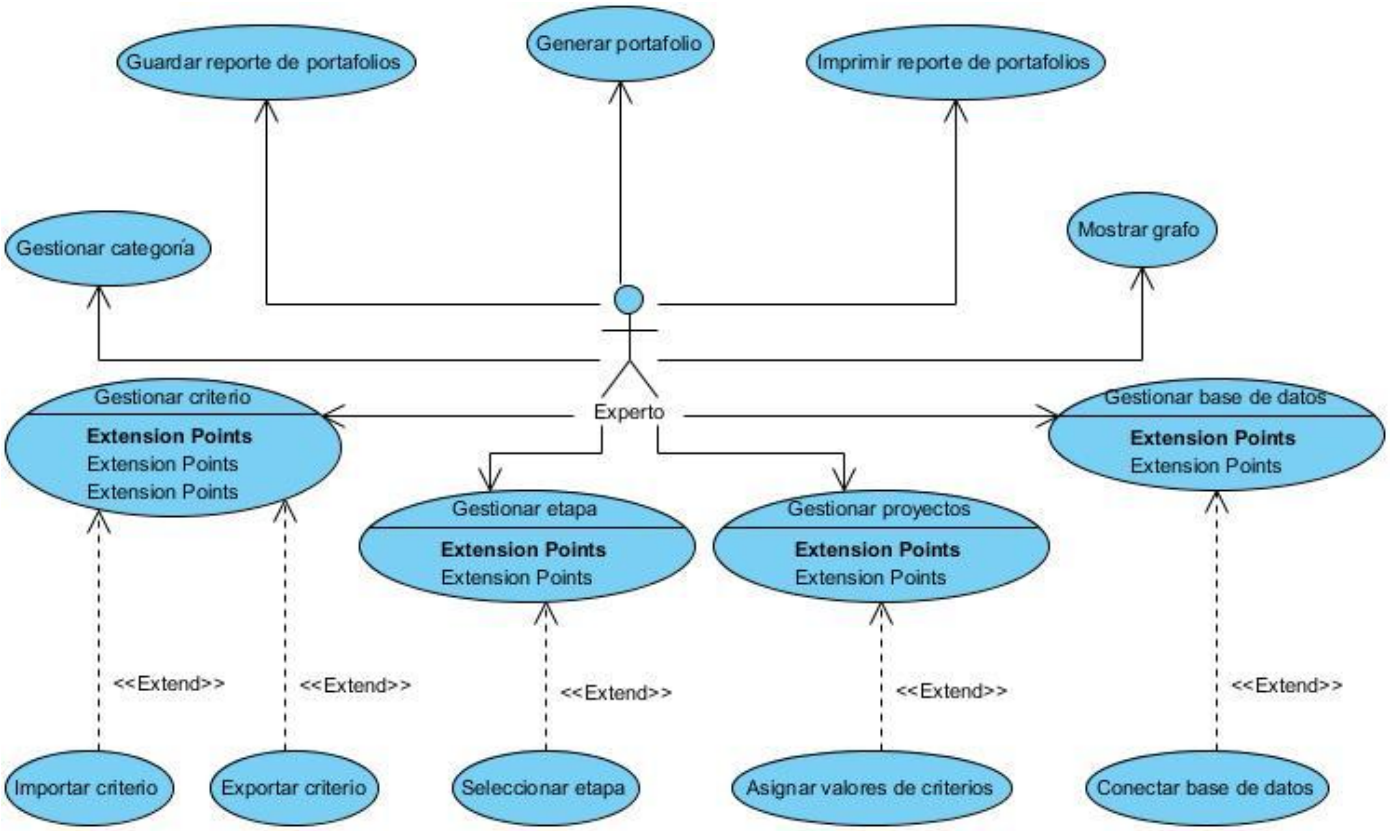


Figura 2: Diagrama de casos de uso del sistema.

Actores

Un actor es alguien o algo, externo al sistema, que de cierta forma interactúa con este. Los actores no son solamente roles que juegan personas, sino también organizaciones, software y máquinas. (Larman, 2001)

En la tabla 4 se reflejan los actores que componen el sistema en desarrollo:

Tabla 4: Actores del sistema.

Actores del sistema	Descripción
Experto	Es el responsable de introducir los datos al sistema y obtener las mejores combinaciones de proyectos.

Listado de casos de uso del sistema

Tabla 5: Gestionar criterio.

CU-1	Gestionar criterio
Actor	Experto
Descripción	El experto puede adicionar, eliminar y editar criterios.
Referencia	RF-5

Tabla 6: Asignar valores de criterios.

CU-2	Asignar valores de criterios
Actor	Experto
Descripción	El experto asigna valores de criterios a los proyectos.
Referencia	RF-1

Tabla 7: Generar portafolios.

CU-3	Generar portafolios
Actor	Experto
Descripción	El experto obtiene las mejores combinaciones de proyectos que desee generar.
Referencia	RF-1, RF-2

Tabla 8: Importar criterios.

CU-4	Importar criterios
Actor	Experto
Descripción	El experto tiene la posibilidad de importar criterios que tenga guardados.
Referencia	RF-4

Tabla 9: Generar reporte.

CU-5	Generar reporte
Actor	Experto
Descripción	El experto puede guardar el reporte de la combinación de proyectos obtenidos.
Referencia	RF-3,RF-7

Tabla 10: Mostrar grafo de criterios.

CU-6	Mostrar grafo de criterios
Actor	Experto
Descripción	El experto visualiza la estructura del grafo de criterio
Referencia	RF-1,RF-2

Tabla 11: Generar espaciación de los portafolios.

CU-7	Generar espaciación de los portafolios.
Actor	Experto
Descripción	El experto visualiza la de los portafolios en el tiempo.
Referencia	RF-3

Tabla 12: Conectar base de datos.

CU-8	Conectar base de datos
Actor	Experto
Descripción	El experto conecta la base de datos con el sistema.
Referencia	RF-8
CU-8	Conectar base de datos

Tabla 13: Gestionar base de datos.

CU-9	Gestionar base de datos
Actor	Experto
Descripción	El experto puede Crear una nueva base de datos o Leer base de datos existente.
Referencia	RF-9
CU-9	Gestionar base de datos

Tabla 14: Gestionar etapa.

CU-10	Gestionar etapa
Actor	Experto
Descripción	El experto puede adicionar, eliminar y editar etapas de trabajo.
Referencia	RF-1, RF-2, RF-3
CU-10	Gestionar etapa

Descripción de casos de uso del sistema

La descripción de casos de uso del sistema se realiza con el objetivo de describir brevemente cómo funcionarán los casos de uso en el sistema. En cada descripción se especifica el propósito general de cada uno de ellos, el resumen de su funcionamiento, así como las condiciones que deben existir para que estos ocurran. A continuación se describe el caso de uso Generar Portafolio, ver tabla 15. Las descripciones textuales correspondientes al resto de los casos de usos se muestran en el Anexo 2: Descripciones de Casos de Usos.

Tabla 15: Descripción del caso de uso Generar portafolios.

Caso de uso	Generar portafolios
Resumen	El caso de uso inicia cuando el experto selecciona la opción Generar portafolios, introduce los datos, el sistema muestra la información y termina el caso de uso.
Referencia	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6
Precondiciones	Tener la base de datos conectada, haber insertado cierta cantidad de proyectos y haber adicionado criterios de selección para cada uno de ellos.
Pos condiciones	Se obtiene las mejores combinaciones de proyectos que deseen generar.
Prioridad	Crítico
Flujo normal de eventos	
Acción del actor	**Acción del sistema**
1. Selecciona la opción Portafolio. 3. Se introducen los datos correspondientes y se escoge la opción Generar Portafolios.	2. Muestra un formulario donde el experto puede introducir los datos. 4. Verifica que los valores de criterios estén llenos. 5. Si los valores de criterio no están llenos, véase flujo alterno 5.1. 6. Si los valores de los criterios están llenos el sistema muestra las mejores combinaciones de proyectos terminando así el caso de uso.
Flujo alterno 5.1	
Acción del actor	**Acción del sistema**
	5.1 Si los valores de los criterios no están llenos el

	sistema muestra un mensaje de error para que se llenen y retorna al paso 1.

3.4 Modelo de Análisis

Una vez identificados los requisitos del sistema y agrupadas las funcionalidades en los casos de uso, es posible obtener una visión más detallada del sistema que especifique qué debe hacer el mismo, lo cual es posible representar a través del Modelo de Análisis.

El Modelo de Análisis constituye la primera aproximación al Modelo de Diseño, es el resultado del análisis de los casos de uso y está conformado por las clases del análisis (interfaz, control y entidad), las cuales encapsulan las diferentes funcionalidades que representan los casos de uso. (Jacobson, et al., 2000)

En este epígrafe se puede encontrar el diagrama del análisis del caso de uso Generar portafolios. Pueden encontrarse los restantes diagramas en el Anexo 3: Diagrama de Análisis.

3.4.1 Realización de caso de uso – análisis

Una realización de caso de uso – análisis es un colaboración dentro del Modelo de Análisis que describe cómo se lleva a cabo y se ejecuta un caso de uso determinado en términos de clases del análisis y de sus objetos del análisis en interacción; proporciona por tanto una traza directa hacia un caso de uso concreto del Modelo de Casos de Uso. (Jacobson, et al., 2000)

La realización de los casos de uso del sistema que se propone incluye diagramas de clases que muestran las clases del análisis y sus relaciones, y diagramas de interacción que explican gráficamente cómo los objetos interactúan a través de mensajes para realizar el flujo de eventos de cada caso de uso.

Diagramas de clases

A continuación se muestra el diagrama del análisis del caso de uso Generar portafolios, ver figura 3. Los diagramas del análisis correspondiente al resto de los casos de usos se muestran en el Anexo 3: Diagramas del Análisis.

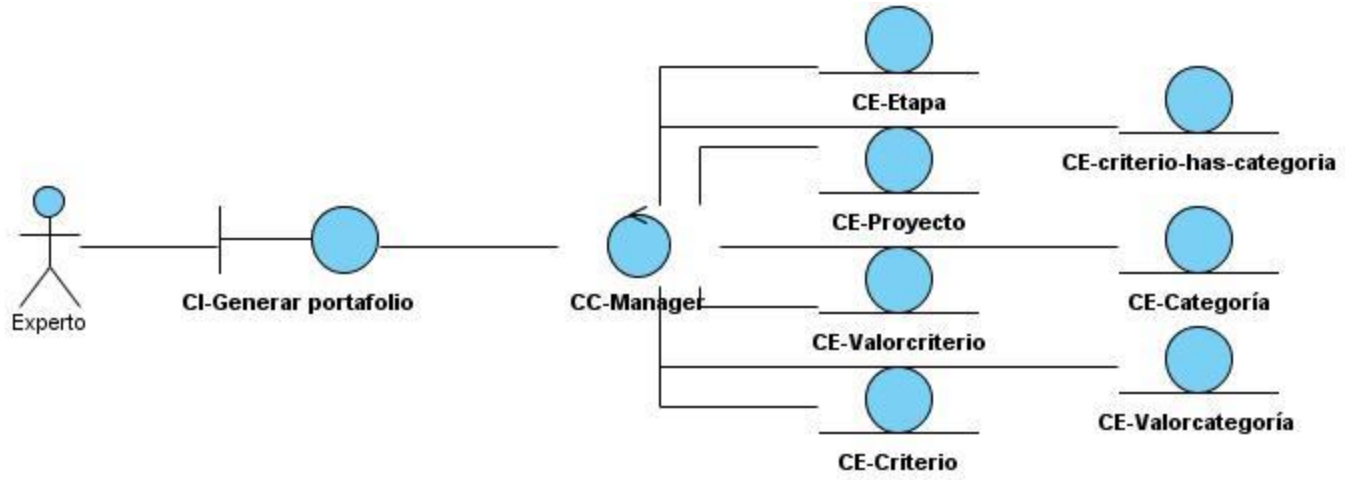


Figura 3: Diagrama de clases del análisis Generar Portafolios.

Diagramas de interacción

En el análisis, las interacciones entre objetos se pueden representar a través de diagramas de secuencia o de colaboración. Para la solución que se propone, se emplean diagramas de colaboración, pues el objetivo fundamental es identificar requisitos y responsabilidades sobre los objetos, y no identificar secuencias de interacción detalladas y ordenadas cronológicamente, para lo cual serían más factibles los diagramas de secuencia.

Un Diagrama de Colaboración (DColaboración) es similar a un diagrama de clases del análisis, pero contiene instancias y enlaces en lugar de clases y asociaciones. Muestra cómo interactúan los objetos, numerando los mensajes que se envían unos a otros. (Jacobson, et al., 2000)

A continuación se muestra el diagrama de colaboración para el caso de uso Generar portafolios, ver figura 4. Los diagramas de interacción correspondiente al resto de los casos de usos se muestran en el Anexo 4: Diagramas de Colaboración.

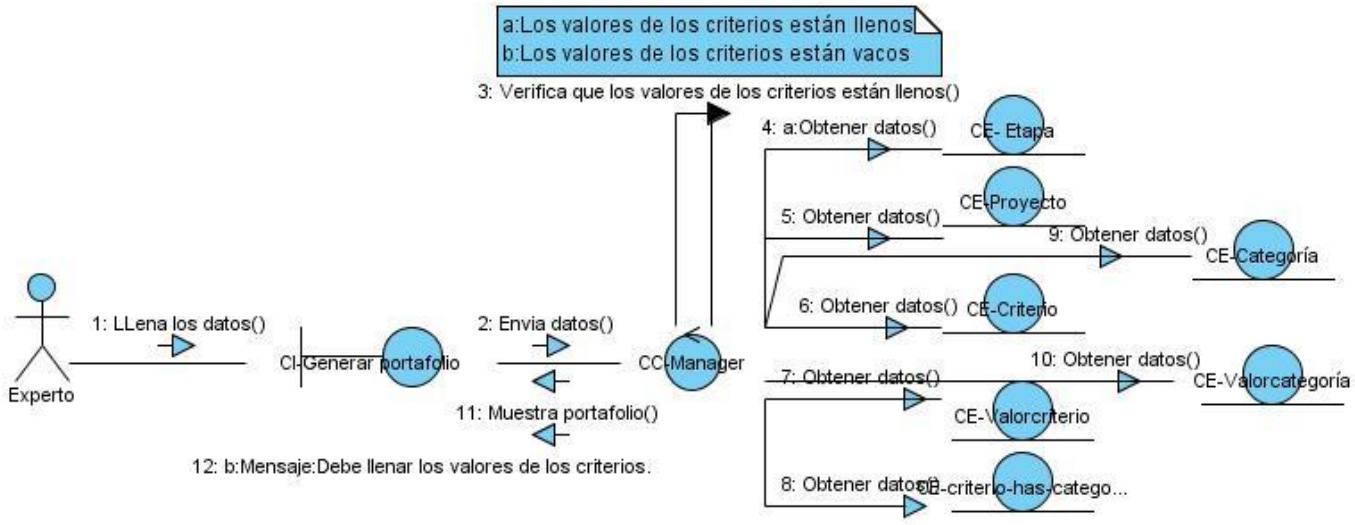


Figura 4: Diagrama de colaboración Generar Portafolios.

3.5 Modelo de Diseño

El Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el Modelo de Diseño sirve de abstracción a la implementación del sistema, por lo que es utilizado como una entrada fundamental de las actividades de implementación.

3.5.1 Realización de caso de uso – diseño

Una realización de caso de uso – diseño es una colaboración en el Modelo de Diseño que describe cómo se realiza un caso de uso específico, y cómo se ejecuta, en términos de clases de diseño y sus objetos; proporciona por tanto una traza directa a una realización de caso de uso – análisis en el Modelo de Análisis. (Jacobson, et al., 2000)

La realización de los casos de uso – diseño del sistema que se propone, incluye diagramas de clases que muestran las clases del diseño participantes y sus relaciones.

Diagramas de clases del diseño

Un diagrama de clases del diseño representa las especificaciones de las clases e interfaces de software en una aplicación. A continuación se muestra el diagrama de clases de diseño para el caso de uso Generar portafolios, ver figura 5. Los diagramas de clases del diseño correspondiente al resto de los casos de usos se muestran en el Anexo 5: Diagramas de Clases del Diseño.

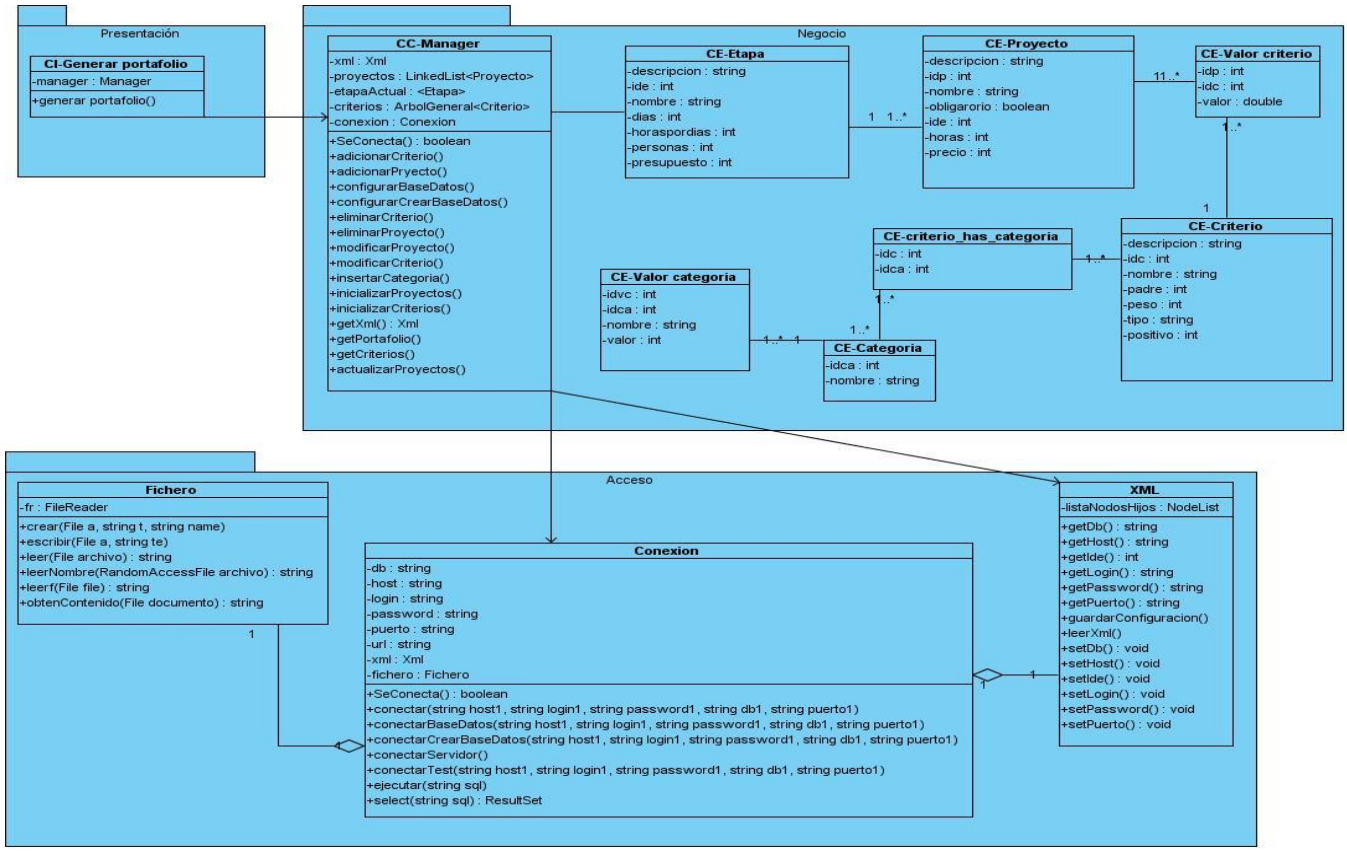


Figura 5: Diagrama de clases de diseño Generar Portafolios.

3.6 Modelo de datos

A partir del diagrama de clases persistentes se obtuvo el siguiente modelo de datos:

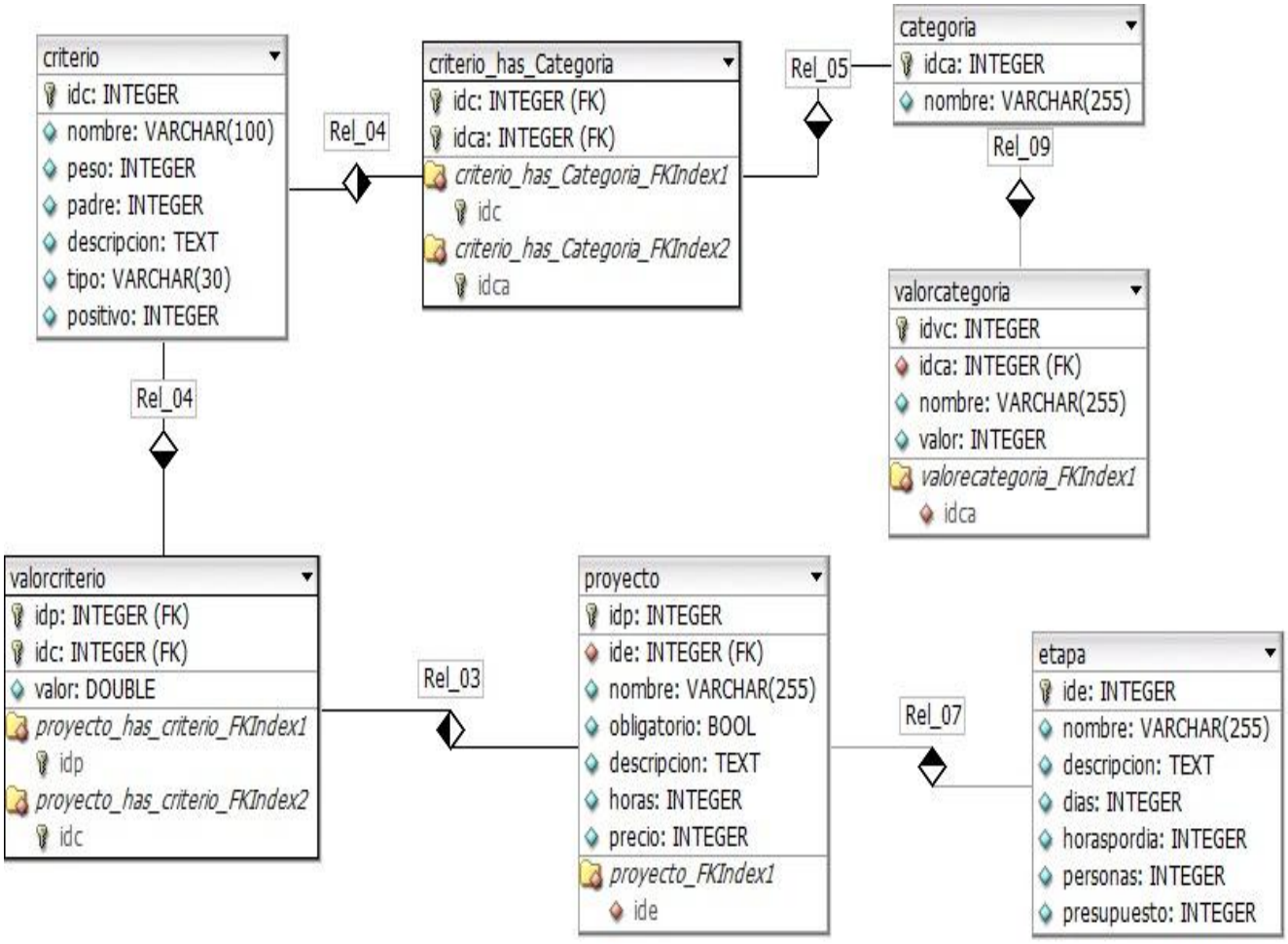


Figura 6: Modelo de datos.

Ver las tablas 5-10, del modelo de datos en el Anexo 6: Modelo de Datos.

3.6.1 Diseño de la base de datos

El diseño de la base de datos es de suma importancia pues debe almacenar información y permitir al experto recuperarla y actualizarla de acuerdo a sus peticiones. Se ofrece el diagrama de clases persistentes y el modelo de datos que dan soporte al contenido manejado por el sistema.

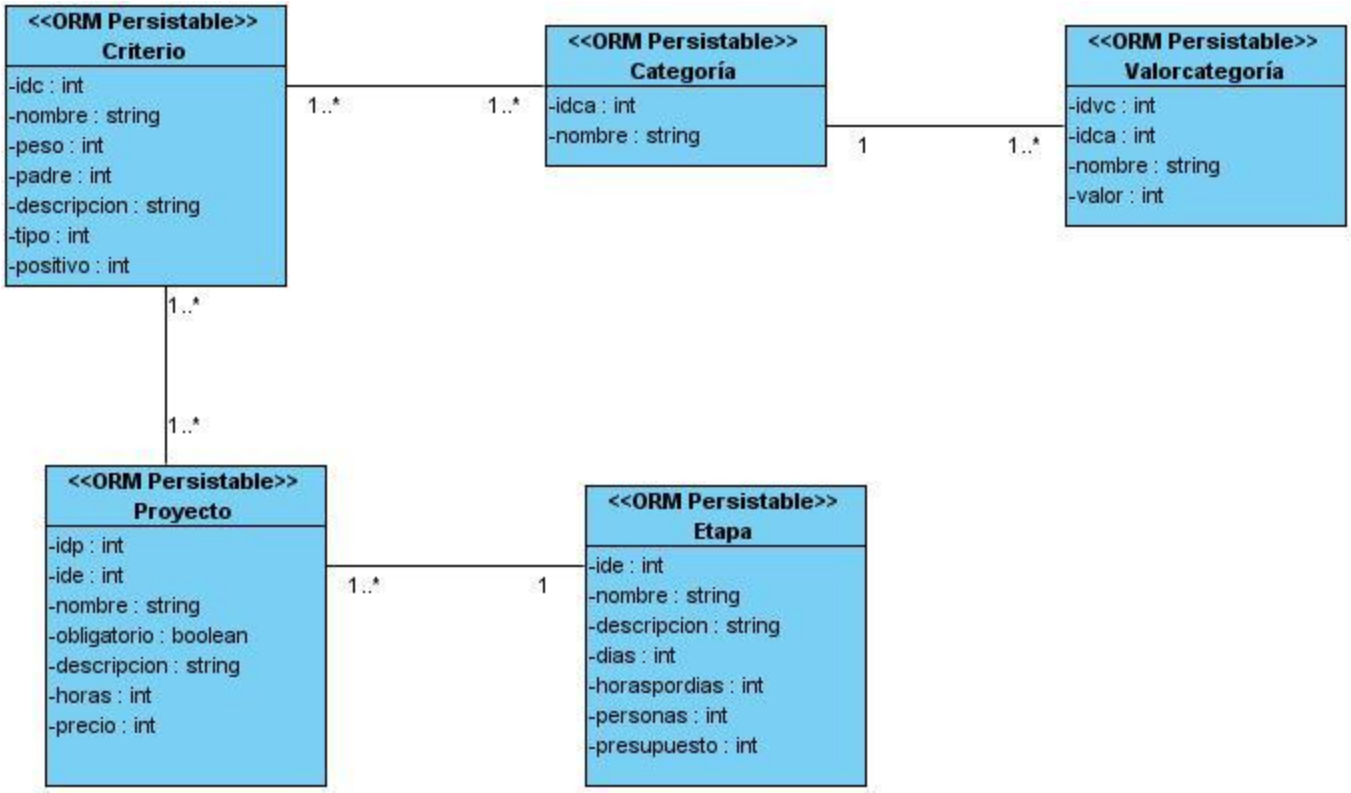


Figura 7: Diagrama de clases persistentes.

Conclusiones

- ✓ En este capítulo se crea el proceso de funcionamiento del sistema a través del modelo del dominio, el cual permitió comprender los procesos básicos del funcionamiento.
- ✓ Se obtuvo un listado de funcionalidades que debe tener el sistema, expresados en los requerimientos funcionales, además se describieron los requerimientos no funcionales que definen las cualidades que el sistema debe cumplir.
- ✓ Quedó evidenciado que la captura de los requisitos es un paso esencial para el análisis del sistema, dando así una visión del futuro de la aplicación.
- ✓ Se elaboraron los diagramas de casos de uso del sistema y se definieron los mismos.

- ✓ Se identificaron los casos de usos críticos para permitir dar prioridad a su realización.
- ✓ Se logró modelar los diagramas de clases del análisis de los casos de usos y los diagramas de interacción para cada escenario de los mismos.
- ✓ Se definió el diseño de la aplicación, esforzándose por conservar la estructura del sistema que sirve como esquema para la implementación.

Capítulo IV Implementación y Prueba del Sistema

Introducción

Partiendo de los resultados del diseño, en el presente capítulo se analiza la implementación, se muestra el diagrama de despliegue y el diagrama de componentes como objetivo primordial de esta fase. Además, se realiza una validación de la solución propuesta mediante las pruebas a las cuales se sometió la aplicación para verificar su completitud.

4.1 Modelo de Implementación

El Modelo de Implementación describe cómo los elementos del Modelo de Diseño, como las clases, se implementan en términos de componentes y subsistemas de implementación. Describe, además, la forma en que se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados. El Modelo de Implementación es la entrada principal de las etapas de prueba que siguen a la implementación. (Booch, et al., 2000)

Diagrama de despliegue

El diagrama de despliegue muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución run-time [14]en las instancias de los nodos de proceso. (Larman, 2001)

El diagrama de despliegue que aparece a continuación muestra la disposición física de los distintos nodos que componen el sistema en tiempo de ejecución, los links de comunicación y las instancias de los componentes y objetos que residen en los mismos. Las conexiones establecidas en este diagrama son asociaciones de comunicación entre los nodos, y se etiquetan con un estereotipo que identifica el protocolo de comunicación o la red utilizada.

[14] Tiempo real

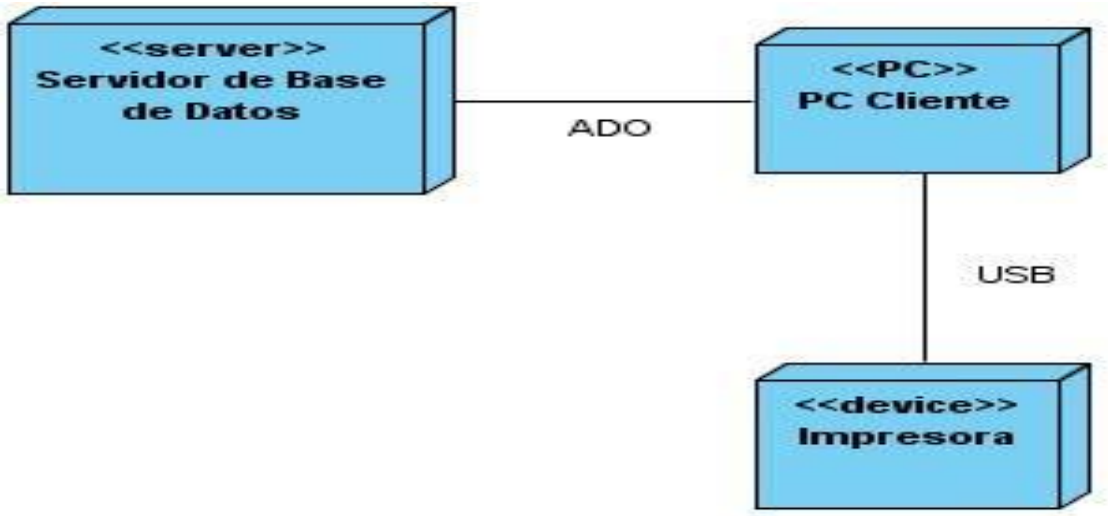


Figura 8: Diagrama de despliegue.

Diagrama de componentes

Por su parte, el diagrama de componentes es otro de los artefactos importantes que incluye el Modelo de Implementación. El mismo muestra elementos del modelo, tales como, los componentes y sus relaciones. Se utiliza para modelar la vista estática del sistema y muestra la organización y las dependencias lógicas entre los componentes de software, sean estos de código fuente, binarios o ejecutables. (Booch, et al., 2000)

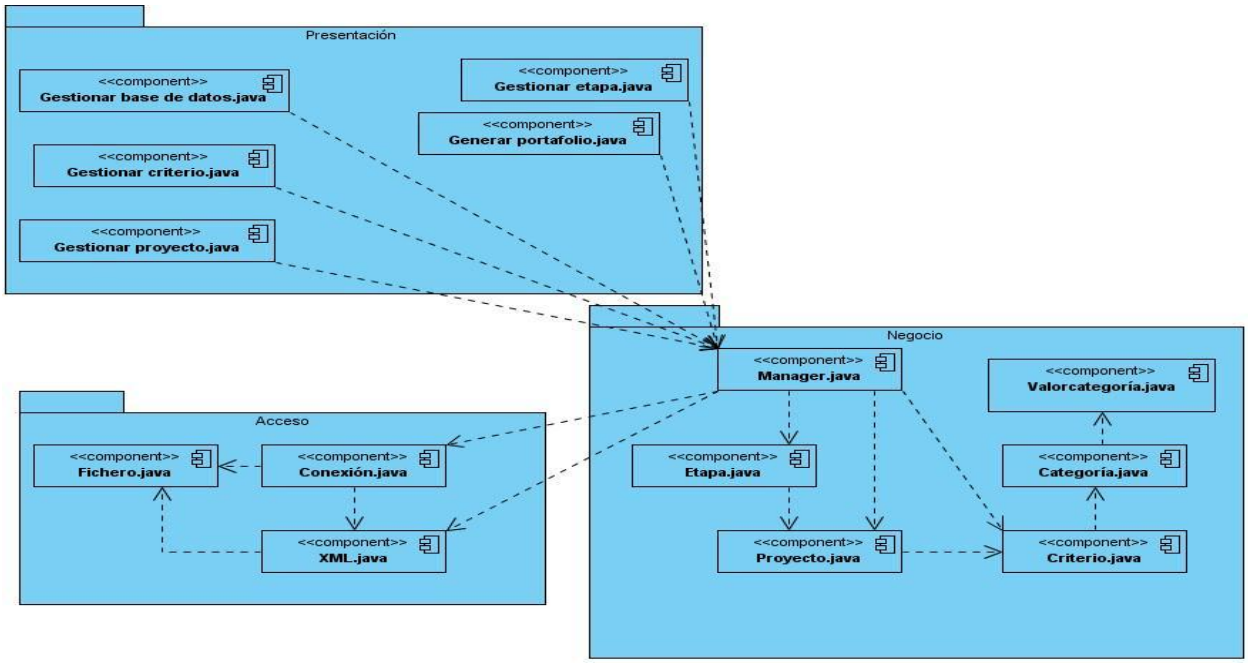


Figura 9: Diagrama de componentes.

4.2 Pruebas de software

Las pruebas de software son un elemento crítico para la garantía de la calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. El objetivo fundamental de las pruebas es descubrir diferentes clases de errores con la menor cantidad de tiempo y de esfuerzo. Aunque las pruebas no pueden asegurar la ausencia de defectos; sí pueden demostrar que existen defectos en el software. (Pressman, 2002)

Una de las últimas fases del ciclo de vida del desarrollo de software es el flujo de trabajo de pruebas, al cual es necesario dedicarle un importante tiempo pues dicha actividad está encaminada a encontrar errores en el software.

La actividad de probar el sistema y verificar que se encuentre libre de defectos tiene muchos beneficios. La calidad es una variable muy importante para todo producto y uno de los caminos para garantizarla es siguiendo esta doctrina, además proporciona una medida del progreso del trabajo que se despliega.

Método de prueba

Con el objetivo de diseñar pruebas que demuestren que cada función es plenamente operacional se hace uso del método de caja negra. Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Las pruebas de caja negra permiten al ingeniero del software derivar conjuntos de condiciones de entrada que ejercitarán completamente todos los requisitos funcionales de un programa. (Pressman, 2005)

Dentro del método de caja negra se hace uso de la técnica Partición Equivalente por ser una de las más efectivas para examinar los valores válidos e inválidos de las entradas existentes en el software. La Partición Equivalente divide el dominio de entrada de un programa en clases de datos a partir de las cuales se derivan casos de prueba (CP). (Pressman, 2005)

4.2.1 Diseños de casos de prueba

Un caso de prueba específica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. (Booch, et al., 2000)

Además los casos de pruebas deben verificar que:

- ✓ Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificaciones de los requerimientos.
- ✓ Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

A continuación se muestran las Pruebas del caso de uso Gestionar base de datos, ver el resto de los

casos de pruebas en el Anexo 7: Casos de Pruebas.

Tabla 16: Prueba del caso de uso Gestionar base de datos escenario "Crear nueva base de datos"

Condición de entrada	Casos válidos	Casos no válidos
Servidor	Nombre del servidor	Campo vacío
Base de datos	Nombre de la base de datos	Campo vacío
Usuario	Cadena de caracteres	Campo vacío o usuario existente
Contraseña	Cadena de caracteres	Campo vacío
Puerto	Número entero	Campo vacío o cadena de caracteres

Caso de uso	**Gestionar base de datos escenario "Crear nueva base de datos"**
Caso de prueba	Permitir crear una nueva base de datos introduciendo correctamente los datos.
Entrada	El usuario introduce los datos de la base de datos correctamente de la forma: Servidor: localhost Base de datos: mysppo Usuario: root Contraseña: root Puerto: 3306
Resultado	El sistema crea una nueva base de datos.
Condiciones	No debe existir ningún campo vacío y los datos deben de ser del tipo especificado.

Caso de uso	**Gestionar base de datos escenario "Crear nueva base de datos"**
Caso de prueba	Crear una nueva base de datos introduciendo mal los datos o dejando campos vacíos.
Entrada	Cualquiera de los campos está vacío o no son los correctos.
Resultado	El sistema muestra un mensaje de error especificando que los datos introducidos no son los correctos o algún campo está vacío.
Condiciones	Deben existir campos vacíos o con datos incorrectos.

4.3 Resultados de las Pruebas

Una evaluación de prueba es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, la cobertura de código y el estado de los defectos. (Booch, et al., 2000)

Durante el transcurso de la etapa de pruebas a la aplicación, se detectaron 4 no conformidades (NC) significativas, 3 no significativas y 1 de recomendación, como se muestra en la tabla 14.

Tabla 17: Resumen de las NC detectadas

Iteraciones	Significativa	No significativa	Recomendación	Total
1	3	2	1	6
2	1	1	0	2
3	0	0	0	0

Conclusiones

- ✓ En este capítulo se expuso la fase de implementación verificando todos los requisitos establecidos.
- ✓ Se representó el diagrama de componentes que muestra la organización y la dependencia entre un conjunto de componentes.
- ✓ Se representó el diagrama de despliegue que describe el modelo físico del sistema.
- ✓ Se provee la vista de implementación del sistema, esta etapa se caracteriza por brindar los resultados visibles ya que queda implementada la aplicación.
- ✓ Se logró desarrollar los casos de prueba para probar el curso fundamental de las funcionalidades.

Conclusiones Generales

La utilización de las tecnologías de la informática y las comunicaciones tiene gran importancia para la agilización y optimización de los procesos llevados a cabo en cualquier organización, empresa o institución.

Se obtienen los siguientes resultados dando respuesta a los objetivos planteados:

- ✓ El análisis de los principales aspectos teóricos permitió obtener un estado del arte de la selección de portafolios de proyectos.
- ✓ Con la creación del modelo matemático se enriqueció la rama de la selección de portafolios de proyectos unificando en un modelo varias técnicas de selección.
- ✓ El análisis y diseño contribuyó con el desarrollo de la aplicación.
- ✓ Se desarrolló un sistema que implemente la solución matemática elaborada en dicho trabajo.
- ✓ Se confeccionó un manual de usuario que brinda información detallada sobre el trabajo con la herramienta.
- ✓ Con la obtención del software se logra un mejor proceso de selección de portafolios de proyectos, aligerando el tiempo de aplicación y permitirá el análisis de criterios que varias veces se obviaban por la complejidad de los mismos y el inmenso cúmulo de cálculos que acarreaban.

Recomendaciones

Una vez terminada la aplicación se puede constatar que los objetivos trazados al comenzar el trabajo fueron resueltos de manera satisfactoria, aunque se debe tener en cuenta que esta no es más que una primera versión, con vista a enriquecer la solución propuesta se sugiere:

- ✓ Continuar con el estudio de la rama de la selección de portafolios de proyectos.
- ✓ Enriquecer el modelo de manera tal que pueda brindar otras funcionalidades.
- ✓ Realizar un manual de usuario más específico.

REFERENCIAS BIBLIOGRÁFICAS

Referencias Bibliográfica

1. **Rodríguez Velázquez, Juan Alberto y Steegmann Pascual, Cristina . 2010.** *Modelos Matemáticos.* 2010.

2. **AplicacionesEmpresariales.com. 2008.** AplicacionesEmpresariales.com. [En línea] 4 de Mayo de 2008. [Citado el: 23 de Febrero de 2011.] http://www.aplicacionesempresariales.com/xampp-el-servidor-web-listo-para-ser-usado.html.

3. **Beck, Kent. 2000.** *Una explicación de la Programación extrema: aceptar el cambio.* s.l. : Addison-Wesley, 2000.

4. **Booch, Grady, Jacobson, Ivar y Rumbaugh, James. 2000.** *El lenguaje unificado de modelado.* s.l. : Addison Weslley, 2000.

5. **Carazo Fernández, Ana, Caballero Fernández, Rafael y Gómez Núñez, Trinidad. 2010.** *Evaluación y clasificación de las técnicas utilizadas por las organizaciones en las últimas décadas, para seleccionar proyectos.* 2010.

6. **Chacón, José Luis. 2011.** Grafos. [En línea] 2011. http://webdelprofesor.ula.ve/ciencias/jlchacon/materias/discreta/grafos.pdf.

7. **Collector, Garbage. 2011.** Garbage Collector. [En línea] 2011. [Citado el: 22 de Marzo de 2011.] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

8. **DBDesigner. 2011.** DBDesigner. [En línea] 2011. [Citado el: 23 de febrero de 2011.] http://dbdesigner.sourceforge.net/.

9. **Eclipse, La Fundacion. 2012.** Eclipse. *Eclipse.* [En línea] 2012. [Citado el: 5 de Mayo de 2012.] http://www.eclipse.org/.

10. **ECURed. 2012.** Proceso_Unificado_de_Desarrollo. [En línea] 21 de Febrero de 2012. [Citado el: 24 de Febrero de 2012.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo..

11. **Ecured. 2012.** Ecured. [En línea] 2012. [Citado el: 12 de Febrero de 2012.] http://www.ecured.cu/index.php/Estilo_arquitectural_en_capas_(N-Layer).

REFERENCIAS BIBLIOGRÁFICAS

12. **fabFORCE.net. 2011.** fabFORCE.net. [En línea] 2011. [Citado el: 7 de Diciembre de 2011.] http://www.fabforce.net/dbdesigner4/.

13. **Finanzas Personales : Gestión de la cartera : cartera de Optimización de 1, 0. 2002.** Finanzas Personales : Gestión de la cartera : cartera de Optimización de 1,0. [En línea] 2002. [Citado el: 10 de noviembre de 2011.] http://translate.googleusercontent.com/translate_c?hl=es&prev=/search%3Fq%3DPortfolio%2BOptimization%2B1.0%26hl%3Des%26sa%3DN%26biw%3D990%26bih%3D593%26noj%3D1%26prmd%3Dimvnsa&rurl=translate.google.com.cu&sl=en&u=http://www.softandco.com/a/6382/portfolio-.

14. **GSInnova. 2010.** *GSInnova.* [En línea] 2010. [Citado el: 9 de Marzo de 2012.] http://www.rational.com.ar/herramientas/roseenterprise.html.

15. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El proceso unificado del desarrollo de software.* Madrid : Addison Wesley, 2000. 84-7829-036-2..

16. **Larman, Craig. 2001.** *UML y Patrones.* 2da . s.l. : Prentice Hall, 2001.

17. **Mijango, Rafael Cuevas. 2010.** Matemáticas para la toma de decisiones. 2010, Unidad 2.

18. **MySQL, PostGreSQL vs. 2011.** PostGreSQL vs, MySQL . *PostGreSQL vs, MySQL .* [En línea] 2011. [Citado el: 10 de Mayo de 2011.] http://www.danielpecos.com/docs/mysql_postgres/x15.html.

19. **NetBeans. 2011.** Netbeans. [En línea] 2011. [Citado el: Febrero de 4 de 2011.] http://netbeans.org/index_es.html..

20. **PostGreSQL, MySQL vs. 2011.** MySQL vs. PostGreSQL. [En línea] 2011. [Citado el: 7 de Diciembre de 2011.] http://danielpecos.com/docs/mysql_postgres/x57.html.

21. **PostGreSQL-es. 2010.** PostGreSQL-es. [En línea] 2 de 10 de 2010. [Citado el: 4 de Mayo de 2011.] http://www.postgresql.org.es/sobre_postgresql.

22. **Pressman, Roger S. 2002.** *Ingeniería de software: Un enfoque práctico, 5ta Edición.* 5ta edición. 2002. ISBN:8448132149.

23. **—. 2005.** *Ingeniería de software: Un enfoque práctico, 6ta Edición.* 2005. 9701054733.

REFERENCIAS BIBLIOGRÁFICAS

24. **Ramos, Juan Sanchez. 2001-2002.** RECURSOS MULTIMEDIALES EN INGENIERÍA DE TRANSPORTE. [En línea] 2001-2002. http://www.material_simulacion.ucv.cl/index.htm.

25. **Sánchez, Raquel Criado. 2010.** *Patrones de secuenciación de actividades en la ensenñanza de inglés como lengua extranjera y su incidencia en el aprendizaje estudio cuasi-experimental.* 2010.

26. **Sierra, María. 2008.** Trabajando con Visual Paradigm for UML. [En línea] 2008. http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf..

27. **slideshare. 2011.** slideshare. [En línea] 2011. [Citado el: 21 de Febrero de 2011.] http://www.slideshare.net/Danto/patrn-mvc.

28. **Tamayo, Ing. Karina Sánchez. 2010.** *Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones.* Ciudad Habana : s.n., 2010.

BIBLIOGRAFÍA CONSULTADA

Biblografía Consultada

1. **Rodríguez Velázquez, Juan Alberto and Steegmann Pascual, Cristina . 2010.** Modelos Matemáticos. 2010.

2. **Andrés, María Mercedes Marqués.** Herramientas CASE. [Online] [Cited: noviembre 4, 2011.] http://www3.uji.es/~mmarques/f47/apun/node75.html.

3. **AplicacionesEmpresariales.com. 2008.** AplicacionesEmpresariales.com. [Online] Mayo 4, 2008. [Cited: Febrero 23, 2011.] http://www.aplicacionesempresariales.com/xampp-el-servidor-web-listo-para-ser-usado.html.

4. **Bass, Len, Clements, Paul y Kazman, Rick. 2003.** *Software Architecture in Practice. Segunda edición.* s.l. : Pearson Education, 2003. Disponible en: http://books.google.com.cu/books?id=mdilu8Kk1WMC&dq=software+architecture+in+practice&printsec=frontcover&source=bn&hl=es&ei=lrG6S7uqJ8P98AaluKGkCA&sa=X&oi=book_result&ct=result&resnum=4&ved=0CCAQ6AEwAw#v=onepage&q=&f=false. . 0-321-15495-9.

5. **Beck, Kent. 2000.** Una explicación de la Programación extrema: aceptar el cambio. s.l. : Addison-Wesley, 2000.

6. **Booch, Grady, Jacobson, Ivar and Rumbaugh, James. 2000.** El lenguaje unificado de modelado. s.l. : Addison Weslley, 2000.

7. **Carazo Fernández, Ana, Caballero Fernández, Rafael and Gómez Núñez, Trinidad. 2010.** Evaluación y clasificación de las técnicas utilizadas por las organizaciones en las últimas décadas, para seleccionar proyectos. 2010.

8. **Carazo, Ana.** [Online] [Cited: Diciembre 10, 2011.] http://www.uv.es/asepuma/recta/ordinarios/9/9_1.pdf..

9. **Chacón, José Luis. 2011.** Grafos. [Online] 2011. http://webdelprofesor.ula.ve/ciencias/jlchacon/materias/discreta/grafos.pdf.

10. **Collector, Garbage. 2011.** Garbage Collector. [Online] 2011. [Cited: Marzo 22, 2011.] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

BIBLIOGRAFÍA CONSULTADA

11. **DBDesigner. 2011.** DBDesigner. [Online] 2011. [Cited: febrero 23, 2011.] http://dbdesigner.sourceforge.net/.

12. **Eclipse, La Fundacion. 2012.** Eclipse. *Eclipse.* [Online] 2012. [Cited: Mayo 5, 2012.] http://www.eclipse.org/.

13. **ECURed. 2012.** Proceso_Unificado_de_Desarrollo. [Online] Febrero 21, 2012. [Cited: Febrero 24, 2012.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo..

14. **Ecured. 2012.** Ecured. [Online] 2012. [Cited: Febrero 12, 2012.] http://www.ecured.cu/index.php/Estilo_arquitectural_en_capas_(N-Layer).

15. **fabFORCE.net. 2011.** fabFORCE.net. [Online] 2011. [Cited: Diciembre 7, 2011.] http://www.fabforce.net/dbdesigner4/.

16. **Finanzas Personales : Gestión de la cartera : cartera de Optimización de 1, 0. 2002.** Finanzas Personales : Gestión de la cartera : cartera de Optimización de 1,0. [Online] 2002. [Cited: noviembre 10, 2011.] http://translate.googleusercontent.com/translate_c?hl=es&prev=/search%3Fq%3DPortfolio%2BOptimization%2B1.0%26hl%3Des%26sa%3DN%26biw%3D990%26bih%3D593%26noj%3D1%26prmd%3Dimvnsa&rurl=translate.google.com.cu&sl=en&u=http://www.softandco.com/a/6382/portfolio-.

17. **Gallón, Rendón Álvaro.** [Online] ftp://jano.unicauca.edu.co/cursos/Maestria/AplicacionesInternet/transp/RUP.pdf..

18. **GarbageCollector. 2004.** GarbageCollector. [Online] Noviembre 1, 2004. [Cited: Enero 10, 2011.] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php..

19. **González, Agustín J. 2002.** Definiciones: conjuntos, grafos, y árboles. [Online] 2002. http://profesores.elo.utfsm.cl/~agv/elo320/01and02/definiciones/setGraphTree.pdf.

20. **González, Elizabeth Reina Meneses. 2010.** Análisis del Módulo de Gestión de Tiempo del Sistema Integral para la Gestión de Portafolios de Proyectos. Ciudad de La Habana : s.n., 2010.

21. **Grafos.** [Online] http://www.upseros.com/fotocopiadora/ficheros/Matematica%20Discreta/apuntes%20de%20teoria%20de%20grafos.pdf..

BIBLIOGRAFÍA CONSULTADA

22. **—. 2010.** Grafos: Primeras definiciones. [Online] 2010. [Cited: Marzo 21, 2011.] http://www.ual.es/~btorreci/tr-grafos.pdf.

23. **GSInnova. 2010.** GSInnova. [Online] 2010. [Cited: Marzo 9, 2012.] http://www.rational.com.ar/herramientas/roseenterprise.html..

24. **Guth, Robert A. 2006.** El software en línea desafía al software de escritorio. [Online] Julio 27, 2006. [Cited: Noviembre 10, 2011.] http://internetaldia.wordpress.com/2006/07/30/el-software-en-linea-desafia-al-software-de-escritorio/..

25. **Hernández León, Rolando Alfredo and Coello González, Sayda. 2002.** El paradigma cuantitativo de la investigación científica. Ciudad de La Habana : Editorial Universitaria, 2002. 959-16-0343-6..

26. **Hérnandez León, Rolando Alfredo and Coello González, Sayda. 2011.** EL PROCESO DE INVESTIGACIÓN CIENTÍFICA. Ciudad Habana : Editorial Universitaria, 2011. ISBN 978-959-16-1307-3.

27. **IEEE-STD. 1998.** ESPECIFICACIONES DE LOS REQUISITOS . 1998.

28. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 2000.** El proceso unificado del desarrollo de software. Madrid : Addison Wesley, 2000. 84-7829-036-2..

29. **Larman, Craig. 2001.** UML y Patrones. 2da . s.l. : Prentice Hall, 2001.

BIBLIOGRAFÍA CONSULTADA

30. **Leyava Abrahantes, Danay and García Armona, Miguel Raúl . 2011.** Diseño e implementación de un módulo para la realización de presentaciones web reusables sobre Moodle. Ciudad de La Habana : s.n., 2011.

31. **Martinto, Pedro Carlos Pérez.** Diseño teórico de la investigación científica. La Habana : s.n.

32. **Méndez, Laura Cecilia. 2010.** Scribd. [Online] Junio 6, 2010. http://es.scribd.com/doc/51838546/TIPOS-DE-PRUEBAS-DE-SOFTWARE..

33. **Metodologías de desarrollo de software**. *Metodologías de desarrollo de software.* [Online] [Cited: mayo 7, 2012.] http://www2.rhernando.net/modules/tutorials/doc/ing/met_soft.html.

34. **Mijango, Rafael Cuevas. 2010.** Matemáticas para la toma de decisiones. 2010, Unidad 2.

35. **MySQL, PostGreSQL vs. 2011.** PostGreSQL vs, MySQL . *PostGreSQL vs, MySQL .* [Online] 2011. [Cited: Mayo 10, 2011.] http://www.danielpecos.com/docs/mysql_postgres/x15.html.

36. **NetBeans. 2011.** Netbeans. [Online] 2011. [Cited: 4 Febrero, 2011.] http://netbeans.org/index_es.html..

37. **Nobrega., Kenia Rondón. 2010.** Análisis del Módulo de Gestión de Costos del Sistema Integral de Gestión de Portafolios de proyectos. Ciudad de La Habana : s.n., 2010.

38. **Pecos, Daniel. 2007.** PostGreSQL vs. MySQL. [Online] 2007. [Cited: Enero 5, 2011.] http://danielpecos.com/docs/mysql_postgres/x15.html..

39. **Pérez , Maikel Y., Vázquez, Leyva and Piñero, Pedro Y.** [Online] [Cited: Noviembre 9, 2010.] http://octi.guanajuato.gob.mx/sinnco/formulario/MT/MT2009/MT9/SESION1/MT91_MLEYVA_145.pdf..

40. **PostGreSQL, MySQL vs. 2011.** MySQL vs. PostGreSQL. [Online] 2011. [Cited: Diciembre 7, 2011.] http://danielpecos.com/docs/mysql_postgres/x57.html.

41. **PostGreSQL-es. 2010.** PostGreSQL-es. [Online] 10 2, 2010. [Cited: Mayo 4, 2011.] http://www.postgresql.org.es/sobre_postgresql.

BIBLIOGRAFÍA CONSULTADA

42. **Pressman, Roger S. 2002.** Ingeniería de software: Un enfoque práctico, 5ta Edición. 5ta edición. 2002. ISBN:8448132149.

43. **—. 2005.** Ingeniería de software: Un enfoque práctico, 6ta Edición. 2005. 9701054733.

44. **Project Management Institute, Inc. 2004.** Guía de los Fundamentos de la Dirección de Proyectos. Tercera Edición. Pennsylvania : Newtown Square, 2004.

45. **Proyectos Ágiles**. [Online] [Cited: noviembre 9, 2011.] http://www.proyectosagiles.org/que-es-scrum.

46. **Ramos, Juan Sanchez. 2001-2002.** RECURSOS MULTIMEDIALES EN INGENIERÍA DE TRANSPORTE. [Online] 2001-2002. http://www.material_simulacion.ucv.cl/index.htm.

47. **Romero, Gladys Marsi Peñalver. 2008.** "Trabajo de diploma: Metodología ágil para proyectos de software libre". *"Trabajo de diploma: Metodología ágil para proyectos de software libre".* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

BIBLIOGRAFÍA CONSULTADA

48. **Sánchez, Raquel Criado. 2010.** *Patrones de secuenciación de actividades en la ensenñanza de inglés como lengua extranjera y su incidencia en el aprendizaje estudio cuasi-experimental.* 2010.

49. **Schmuller, Joseph. 2000.** *Aprendiendo UML en 24 horas.* México : Princice Hall, 2000. 968-444-463-X.

50. **Serrano, Yanitza Rodríguez. 2010.** *Análisis del Módulo de Gestión de Alcance del Sistema Integral de Portafolios de Proyectos.* Ciudad de la Habana : s.n., 2010.

51. **Sierra, Alejandro. 2003.** [Online] Abril 2003. [Cited: Febrero 4, 2011.] http://www.programacionextrema.org/articulos/newMethodology.es.html..

52. **Sierra, María. 2008.** Trabajando con Visual Paradigm for UML. [Online] 2008. http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf..

53. **slideshare. 2011.** slideshare. [Online] 2011. [Cited: Febrero 21, 2011.] http://www.slideshare.net/Danto/patrn-mvc.

54. **Tamayo, Ing. Karina Sánchez. 2010.** Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones. Ciudad Habana : s.n., 2010.

55. **Trabajos Investigativos sobre Sistemas,Softwares,Tecnología Informática entre otros.** [Online] [Cited: nobiembre 3, 2011.] http://indira-informatica.blogspot.com/2007/09/qu-es-mysql.html.

56. **UML tool, business, process modeler and database designer for software development team.** UML tool business, process modeler and database designer for software development team. [Online] [Cited: octubre 24, 2011.] http://www.visual-paradigm.com.