Universidad de las Ciencias Informáticas

Facultad 2

Análisis y Diseño de un Sistema Inteligente de clasificación de Proyectos Informáticos para la Universidad de Ciencias Informáticas.

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

Autor(es):

Katia Reyes Moreira.

Lendys Daimy Pindado Delgado.

Tutor(es):

MSc. Yadira Ruiz Constanten.

Ing. Dasiel Cordero Morales.

Junio, 2012

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del presente trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Katia Reyes Moreira	Lendys Daimy Pindado Delgado
Autor	Autor
MSc. Yadira Ruiz Constanten	Ing. Dasiel Cordero Morales.
Tutor	Co-tutor

AGRADECIMIENTOS

Le agradezco a Dios por haberme dado la oportunidad de que en la realización de mi tesis haya podido contar con el apoyo de personas tan especiales como:

Mi sobrina que día a día con su alegría y cariño me motiva y ayuda.

Mi hermana de la que he recibido un apoyo incondicional.

Mi abuela que ha sido mi guía y ejemplo, que siempre me ha tenido presente en sus oraciones.

A mi cuñado por su optimismo y perseverancia.

Mis amigas Magalys, Anamarys, Noradys, Yoania y al resto de las mimis, que aun estando lejos influyeron positivamente sobre mi preparación.

Lendys

AGRADECIMIENTOS

A mi mami por confiar siempre en mí y darme su apoyo incondicional en todo momento, por haberme conducido siempre por el camino correcto.

A mi Abu (mi otra mami) por ser mi bujía inspiradora, siempre velando por mi futuro.

A mi papá por ser mi ejemplo a seguir y por enseñarme que no hay meta que no alcance el esfuerzo y el empeño.

A mi tiota y mi tiote por ayudarme y apoyarme siempre en todo, por quererme como una hija.

A mi familia en general por siempre estar pendiente de mí.

A mi novio por tener paciencia conmigo, brindarme apoyo y aliento durante estos años, especialmente durante el desarrollo de este trabajo.

A mis amistades de la Universidad, especialmente a Yudi, Prada y Osmany. A todos mis compañeros de aula, de la misión, de los proyectos, a las muchachitas con las que he convivido; gracias por brindarme su amistad, por estar en las buenas y en las malas, por convertirse en mi familia.

Al profesor Reyder que a pesar de no conocerme me ayudó y se convirtió en un tutor más.

A la profesora Yadira por darnos la oportunidad de hacer este sueño realidad, por ayudarnos cuando más lo necesitábamos. A Dasiel por guiarnos a lo largo del trabajo

A todas aquellas personas que de una forma u otra han colaborado para la realización de este trabajo.

Katia

Esta tesis va dedicada a la persona más especial de mi vida, mi madre, ejemplo de amor, dedicación, ternura y consagración, a la cual le estaré eternamente agradecida.

Para usted Nivia Delgado Hernández

MI VIDA.

Lendys

Dedico este trabajo a mi mami y mi papi, a mi abu, a mi tiota, a mi tiote, a mis hermanis y a miketu, por brindarme su amor incondicional, su confianza, por apoyarme en todo momento.

A todos ellos, por tomar este sueño como parte de los suyos, porque sé que un triunfo mío es orgullo y victoria de ellos también.

Katia

RESUMEN

El desarrollo de proyectos informáticos ha alcanzado gran auge en la actualidad. Los mismos son empleados para dar solución a una gran cantidad de problemas. En la Universidad de Ciencias Informáticas (UCI) el desarrollo de software se encuentra entre sus pilares fundamentales, trayendo consigo la existencia de una gran cantidad de proyectos informáticos. Actualmente la UCI no cuenta con algún mecanismo o sistema que recoja la experiencia y conocimiento adquirido en el desarrollo de los proyectos, que posteriormente pueda ser utilizada para la estimación del tiempo de duración de los mismos. Agruparlos haciendo uso de algún criterio de clasificación sería una manera de obtener elementos que los mismos tengan en común, que contribuyan a la estimación de su duración. En la UCI los proyectos se encuentran organizados según criterios establecidos por la Dirección General de Producción, sin embargo, no brindan los elementos necesarios que permitan realizar la correcta estimación de los mismos.

En este trabajo se realizará un análisis de los criterios de clasificación de los proyectos de desarrollo de software, para posteriormente utilizar el más idóneo. Se hará uso de las facilidades que brinda el Razonamiento Basado en Casos en tareas de clasificación, ya que simula el comportamiento de un experto humano para resolver problemas en un dominio específico, ayudando al proceso de toma de decisiones. El uso de esta técnica posibilitará la obtención de resultados más acertados y así lograr valores de estimación más reales.

PALABRAS CLAVES

Base de Casos, Proyecto, Razonamiento Basado en Casos.

Contenido

INTRODUCCIÓN

La Informática es una ciencia surgida ante la necesidad de controlar y manejar el enorme flujo de información existente. Definida como ciencia aplicada, que abarca el estudio y aplicación del tratamiento automático de la información, utilizando dispositivos electrónicos y sistemas computacionales; también está definida como el procesamiento automático de la información (1). Con el avance de la ciencia y la tecnología, junto a la necesidad del perfeccionamiento de los procesos, el hombre ha ido evolucionando en su forma de pensar y de dar soluciones a las problemáticas a las que se enfrenta. El progreso de la informática ha favorecido el desarrollo de soluciones informatizadas para los procesos de la vida cotidiana, lo que ha propiciado el aumento acelerado de la cantidad de proyectos informáticos.

Un proyecto es esencialmente un conjunto de actividades interrelacionadas, donde intervienen las personas, con un inicio y una finalización definida, que utiliza recursos limitados para lograr un objetivo deseado (2). Lo que diferencia a los proyectos informáticos de los demás proyectos existentes es la obtención del software, el cual constituye una producción inmaterial del cerebro humano y posiblemente una de las estructuras más complicadas que la humanidad conoce. Este tipo de proyecto constituye un elemento fundamental de modernización, y los grandes avances científicos-tecnológicos no se conciben sin su continua participación.

En la vida diaria clasificar resulta una tarea esencial, pues sirve para organizar y ordenar todo lo que rodea al hombre. Mantener estructurado y clasificado el trabajo que realiza, le permite comprender, controlar y poder elegir sus acciones para resolver situaciones de la vida en sus diferentes contextos. Clasificar los proyectos permitiría gestionar el conocimiento existente en cada uno de ellos, constituyendo una base para la toma de decisiones apoyadas en las experiencias productivas de proyectos similares. En la actualidad existen disciplinas capaces de modelar o simular el pensamiento humano y los procesos que ocurren en él. A la rama de las Ciencias de la Computación dedicada al desarrollo o uso de los ordenadores, con los que se intenta reproducir los procesos de la inteligencia humana se le denomina Inteligencia Artificial (IA) (3).

La IA incluye una disciplina denominada Ingeniería de Conocimiento (IC) que proporciona los métodos y técnicas para construir sistemas computacionales denominados Sistemas Basados en Conocimiento (SBC) (4). Un Sistema Basado en el Conocimiento (SBC) es un sistema capaz de soportar una

representación explícita del conocimiento en algún dominio de interés determinado y de aprovecharlo mediante los mecanismos de razonamiento apropiados para encontrar un alto rendimiento en la resolución de problemas (5). El razonamiento basado en experiencias pasadas es un procedimiento que los seres humanos emplean con mucha frecuencia para solucionar problemas, tanto en la vida diaria como en situaciones en que debe aplicarse conocimiento especializado. Los sistemas que solucionan problemas por analogía con otros se llaman Sistemas de Razonamiento Basado en Casos (SRBC) (6). Un sistema de razonamiento basado en casos (SRBC) es básicamente un modelo de razonamiento que permite resolver problemas, entender situaciones y aprender (4).

El Razonamiento Basado en Casos es un método que cuenta con un alto grado de conocimiento y que recoge experiencias pasadas, además de mejorar los métodos existentes para solucionar problemas y aprovechar la capacidad de aprendizaje de las máquinas. Estos proporcionan un acercamiento a la resolución de problemas, así como un modelo cognoscitivo que representa la forma en que los humanos aprenden y recuerdan.

La Universidad de Ciencias Informáticas (UCI) desde sus inicios ha tenido como uno de sus propósitos convertirse en líder en la producción de productos de software y soluciones informáticas en el país. La Dirección General de Producción (DGP) coordina esta actividad, y brinda servicios para asegurar el correcto desarrollo y terminación de proyectos, productos o servicios. La producción se concentra en el desarrollo de proyectos, en los que se desarrolla software, los mismos se encuentran organizados según los criterios establecidos por dicha Dirección. Uno de los criterios que utiliza es el área temática, entre los que se encuentran la bioinformática, la telemática, la seguridad ciudadana, la informática jurídica, la geoinformática, entre otras (7).

De la misma forma que existe gran diversidad de proyectos de desarrollo de software, existen innumerables clasificaciones para estos, de acuerdo a diferentes criterios de clasificación, como el tamaño, la tecnología a usar, la duración, el bien o el servicio que genera. La DGP también clasifica los proyectos según el tipo de cliente, en nacional, de exportación, de desarrollo interno y proyecto propio. Para una misma área o cliente se pueden desarrollar diferentes tipos de aplicaciones, como por ejemplo aplicaciones de multimedia, sistemas de gestión, entre otros. Pero que se encuentren agrupados según estos criterios no significa que la cantidad de tiempo empleado para su desarrollo sea el mismo, por lo que se puede afirmar que estos criterios no proporcionan los elementos suficientes para posteriormente

realizar una estimación de la duración de los proyectos de desarrollo de software, que permita obtener resultados más reales si se trabaja sobre proyectos similares, que se asemejen en el producto que generan.

Teniendo en cuenta la situación anteriormente expuesta, se plantea como **problema a resolver**: ¿Cómo contribuir a la clasificación de los Proyectos de Desarrollo de Software para facilitar la estimación de su duración?

Según el problema identificado, se define como **objeto de estudio**: Clasificación de proyectos de Desarrollo de Software y el **campo de acción** lo constituyen los sistemas inteligentes que contribuyan a la clasificación de Proyectos de Desarrollo de Software en la Universidad de Ciencias Informáticas, teniendo en cuenta el producto final que se genera.

Para dar solución al problema planteado se define como **objetivo general de la investigación** diseñar un sistema inteligente que contribuya a la clasificación de Proyectos de Desarrollo de Software, teniendo en cuenta el producto final que se genera mediante la utilización del Razonamiento Basado en Casos.

Para dar cumplimiento al objetivo general de la investigación, se definen los siguientes **objetivos específicos**:

- Definir características fundamentales que permitan la clasificación de Proyectos de Desarrollo de Software.
- Diseñar un modelo computacional donde se apliquen técnicas de inteligencia artificial en la consideración de la clasificación de proyectos como entidad principal.
- Definir requisitos funcionales y no funcionales a ser resueltos con el sistema a desarrollar.
- Realizar análisis y diseño del sistema.

Para dar cumplimiento a los objetivos planteados han sido definidas las siguientes **tareas de investigación**:

- Elaboración del marco teórico de la investigación.

- Caracterización de la situación existente en el mundo y en el país sobre la clasificación de proyectos de desarrollo de software, la inteligencia artificial y la técnica Razonamiento basado en casos para definir la posición de los investigadores.
- Estudio de las diferentes técnicas de Inteligencia Artificial que basadas en la acumulación de experiencias y estándares definidos permitan seleccionar la que se utilizará para gestionar clasificación de Proyectos de Desarrollo de Software.
- Análisis de la técnica seleccionada para aplicar sus componentes a la clasificación de Proyectos de Desarrollo de Software.
- Análisis de la metodología de desarrollo, lenguaje de modelado, herramienta case, lenguaje de programación, framework y gestor de base de datos propuesto para el posterior desarrollo del sistema.
- Elaboración de los artefactos correspondientes al análisis y diseño del sistema.

Para el desarrollo de las tareas científicas se utilizan **Métodos de Investigación** en la búsqueda y procesamiento de la información. Los mismos se dividen en teóricos y empíricos. Los **Métodos Teóricos** son factibles en el estudio de las características poco observables del objeto de investigación. Dentro de este grupo se utilizaron:

- El método **Análisis Histórico-Lógico**, posibilitó estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo. El método permitió realizar la primera parte de la investigación, al hacer un análisis bibliográfico de los Sistemas Inteligentes, el Razonamiento Basado en Casos y técnicas de IA más utilizadas en el desarrollo de Sistemas Inteligentes. Dio paso a la exploración de trabajos realizados en el campo de los Sistemas Inteligentes y de soluciones previas existentes a problemas similares al actual. Se utilizó para determinar a través de la evaluación de la bibliografía conceptos de esta temática, que permiten conocer el estado de la evolución actual del fenómeno e identificar posibles mejoras y alternativas de solución.
- El **Analítico-Sintético**, se utilizó para el estudio a partir de fuentes bibliográficas seguras de los Sistemas Inteligentes, los Sistemas de Razonamiento Basados en Casos; así como las clasificaciones de los proyectos de software. Permitió además descomponer el problema de

investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

Los **Métodos Empíricos** tienen gran importancia ya que permiten efectuar el análisis preliminar de la información, así como verificar y comprobar las concepciones teóricas. Dentro de este grupo se utiliza:

- El método de **Entrevista**: Apoyará a la incorporación de conocimientos mediante las entrevistas planificadas efectuadas a los especialistas tanto en Inteligencia Artificial como en Gestión de Proyectos. Se seleccionó una muestra intencional no probabilística. Las entrevistas serán dirigidas a especialistas con gran conocimiento en el tema, como es el caso de líderes de proyecto, directores de centro, especialistas de la dirección de producción, entre otros, dentro de la UCI, sin tener en cuenta el por ciento que representan los mismos dentro de la población.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Introducción

El objetivo fundamental del presente capítulo es abordar los aspectos más relevantes que conforman el fundamento teórico de la investigación. Se describen los conceptos principales. Se realiza el estudio de algunos de los sistemas existentes a nivel mundial y nacional vinculados al problema y que pueden servir como referencia para la elaboración de esta solución.

1.1 Software

Los expertos en computación aún no entienden del todo cómo funciona el software, su comportamiento, sus paradojas y sus límites. Básicamente, el software es un plan de funcionamiento para un tipo especial de máquina, una máquina virtual o abstracta. Una vez escrito mediante algún lenguaje de programación, el software se hace funcionar en ordenadores, que temporalmente se convierten en esa máquina para la que el programa sirve de plan. El software permite poner en relación al ser humano y a la máquina, y también a las máquinas entre sí. Sin ese conjunto de instrucciones programadas, los ordenadores serían objetos inertes, sin capacidad siquiera para mostrar algo en la pantalla (8).

1.2 Proyecto

Un proyecto es la célula básica para la organización, ejecución, financiamiento y control de actividades, recursos materiales y humanos, para lograr en un tiempo determinado los objetivos propuestos. Es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. Podría definirse a un proyecto como el conjunto de las actividades que desarrolla una persona o una entidad para alcanzar un determinado objetivo. Estas actividades se encuentran interrelacionadas y se desarrollan de manera coordinada (9).

El término proyecto es empleado en muchos contextos y su interpretación depende en gran medida del significado que le da cada interlocutor. Esto radica en la ambigüedad que el término refleja al no estar

acompañado de un adjetivo que con certeza clarifique a qué tipo de proyecto se está refiriendo. Existen varios tipos de proyectos dependiendo de los distintos aspectos que se tengan en cuenta con relación al ámbito de desarrollo y la perspectiva que se adopte en un determinado trabajo, como por ejemplo proyectos comunitarios, culturales, logísticos, proyectos de desarrollo de software, etc.

1.3 Proyecto de Desarrollo de Software

Los Proyectos de Desarrollo de Software o Proyectos Informáticos como se le denomina usualmente, tuvieron su origen después de la Segunda Guerra Mundial y se han convertido en un factor clave para el éxito de las empresas. Desarrollar software lleva implícito la combinación de un conjunto de procesos, técnicas y herramientas que propician la obtención del producto software, radicando aquí su carácter innovador. Un proyecto informático puede definirse como un sistema de cursos de acciones simultáneas y/o secuenciales que incluye personas, equipamientos de hardware, software y comunicaciones, enfocadas en obtener uno o más resultados deseables sobre un sistema de información.

Los proyectos informáticos tienen características similares a un proyecto genérico, pero también tienen ciertas peculiaridades. Estos se llevan a término para solucionar un problema perfectamente identificado, por lo que tienen un objetivo definido, concreto y tangible. Son únicos y diferentes de otros proyectos, pero tienen ciertos aspectos que permiten usar analogías con otros. El cambio tecnológico en este entorno es mucho más rápido que en el resto (equipamiento, programas, infraestructuras, etc.).

1.3.1 Clasificación de los proyectos

Básicamente, la clasificación implicará la búsqueda de aquellas cosas que guarden o compartan algún tipo de relación para así agruparlas. Generalmente, el objetivo primordial de la clasificación es encontrar la mejor clasificación posible, para que llegado el momento de la búsqueda de determinada cosa que se clasificó sea más fácil de encontrar, ese es primordialmente el fin de toda clasificación.

En el ámbito de la informática, puede pensarse en una gran variedad de clasificaciones propias para los proyectos. La mayoría de estas se basan principalmente en los diferentes aspectos que se quieran obtener al concluir la agrupación. En la actualidad existen numerosas clasificaciones ya sean por: el grado de libertad, el tamaño de los proyectos, su diseño, el riesgo en la ejecución y grados de libertad en la

implantación, las metodologías de programación, la finalidad de la empresa u organismo que requiera del sistema, entre otros infinitos criterios expuestos por numerosos autores. A continuación se muestra como se clasifican los proyectos de acuerdo con algunos de los criterios mencionados anteriormente.

El tamaño de los proyectos

Este está dado de acuerdo a la cantidad de personas y recursos que interactúan para su realización (10).

- **Proyectos Pequeños**: consisten solamente en implementación. No tienen costos indirectos importantes.
    - Menos de un año de tiempo de desarrollo.
    - Menos de 25 meses-persona de esfuerzo total.
    - Menos de 3 personas en el equipo de trabajo.
- **Proyectos Medianos:** poseen implementación, costos indirectos importantes.
    - De 1 a 2 años de tiempo de desarrollo.
    - Entre 25 y 100 meses-persona.
    - Más de 3 personas en el equipo de trabajo.
- **Proyectos Grandes:** poseen implementación, gerencia de proyecto, control de la calidad, capacitación de personal, hay plan de mantenimiento, hay documentación importante para el uso interno y externo y se genera información para mercadeo.
    - Más de 3 años de tiempo de desarrollo.
    - Más de 100 meses-persona.
    - Más de 10 personas en el equipo de trabajo.

El diseño

- **No estructurados**: Es cuando el diseño es claro y específico, elaborados en forma intuitiva, generalmente por los propios alumnos.
- **Semi-estructurados**: Estos siguen un formato específico, pero dejan la posibilidad de realizar cambios, redefinir metas y objetivos durante su desarrollo.

- **Estructurados:** Tienen un diseño claro y específico, que no da lugar a cambios ni modificaciones. El desarrollo de aplicaciones estructuradas se podría considerar el tipo de proyecto informático más "clásico", siendo además el más conocido y, por tanto, del que existe mayor información y más experiencias. Consiste básicamente en la construcción a medida de una solución software que satisfaga unos determinados requerimientos de usuario, siguiendo el paradigma tradicional, habitualmente denominado "en cascada" debido a que cada fase se ejecuta a continuación de otra (11).

Finalidad de la empresa

- **Inversión privada**: Son proyectos que buscan generar rentabilidad económica y obtener ganancias en dinero (12).
- **Inversión pública o social**: Son los proyectos que buscan alcanzar un impacto sobre la calidad de vida de la población, los cuales no necesariamente se expresan en dinero.

Metodologías de programación

- **Programación clásica**: ciclo de vida en cascada.
- **Programación orientada a objetos:** Este tipo de proyecto también tiene como objetivo la construcción a medida de una aplicación, pero en este caso, aplicando el paradigma más moderno de la "orientación a objetos" (13). Como características esenciales tiene:
    - Proceso **Iterativo** en el sentido de que conlleva el refinamiento sucesivo, por el cual se aplica la experiencia y resultados de cada versión a la siguiente iteración del análisis y el diseño.
    - Proceso **incremental** en el sentido de que cada pasada por un ciclo análisis/diseño/evolución lleva a refinar gradualmente el software.

Tipo de cliente

- **Nacional:** es cuando el producto está destinado a un cliente del país.
- **Extranjeros:** para un cliente extranjero.

Tras el análisis de las clasificaciones anteriores se llega a la conclusión de que las mismas no aportan los elementos necesarios para poder lograr una adecuada estimación del tiempo de duración de los proyectos de desarrollo de software. Además, es evidente que al clasificarlos por estos criterios se podrían encontrar proyectos con soluciones similares y tener clasificaciones diferentes, por lo que se propone realizar la clasificación teniendo en cuenta el producto final que genera el proyecto. Para realizar el proceso de clasificación es importante tener en cuenta el origen de lo que se quiere clasificar, la forma de representar los elementos de dicho dominio, entre otros elementos que hacen que esta tarea se vuela compleja. Una forma de resolver problemas de este tipo es mediante la imitación del comportamiento de un experto en un determinado tema. Además de que se hace uso de la información o el conocimiento del dominio para dar solución a problemas futuros. En este caso el dominio lo constituyen las clasificaciones de los proyectos desarrollados, utilizando estas como una guía para la toma de decisiones. Por esta razón se propone el uso de la Inteligencia Artificial, ya que esta provee los métodos y mecanismos para resolver problemas del mundo real para los cuales el enfoque algorítmico tradicional no es suficiente.

1.4 Inteligencia Artificial

La Inteligencia Artificial (IA) se remonta a las aspiraciones de filósofos como Ramón Llul, Descartes y Leibniz que tenían como objetivo supremo la mecanización total del razonamiento humano, o ingenieros soñadores como Babbage que proyectaron la construcción de máquinas inteligentes. El término "inteligencia artificial" fue acuñado formalmente en 1956 durante la conferencia de Darthmounth dirigida por McCarrthy, Minsky, Rochester y Shannon, donde todos participaban de una metodología en común: el uso del ordenador como analogía fructífera de cara a comprender y simular el comportamiento inteligente de las personas (14).

Se puede definir la Inteligencia Artificial como una ciencia que tiene como objetivo el diseño y construcción de máquinas capaces de imitar el comportamiento inteligente de las personas. Una rama de la Informática que investiga y produce razonamiento por medio de máquinas automáticas y que pretende fabricar artefactos dotados de la capacidad de pensar (14).

La inteligencia artificial (IA) se enfocó inicialmente en la producción de sistemas con la capacidad de asistir al ser humano en la toma de decisiones o en la búsqueda de soluciones a problemas. Además, se

orientó al desarrollo de mecanismos que facilitaran la comunicación con el computador en lenguaje natural, trayendo consigo el surgimiento de los sistemas expertos, actualmente conocidos como SBC (15).

1.4.1 Sistemas basados en el conocimiento (SBC).

Un SBC puede definirse como un sistema computarizado que utiliza conocimiento específico de un dominio y se emplea para dar solución a problemas en dicho dominio. En los mismos se almacena la experticia humana. Estos sistemas son un modelo computacional formado por tres componentes esenciales: la base de conocimientos (BC), en la cual se almacena el conocimiento necesario y las experiencias de los expertos para resolver los problemas del dominio; el motor de inferencia (MI), que se encarga de realizar los procesos de inferencia entre la información contenida en la base de datos o memoria de trabajo y la base de conocimiento, con el fin de obtener las conclusiones que sean necesarias; y la interfaz de usuario, que es la parte del sistema experto que permite la comunicación entre el usuario y el motor de inferencias, además de que permite introducir la información que necesita el sistema y comunica las respuestas del sistema experto al usuario (15).

El conocimiento representado en los SBC es el de los expertos en el dominio, haciendo uso de las experiencias pasadas. Estas experiencias representan conocimiento informal o atajos que permiten al experto encontrar rápidamente solución a un problema, sin tener que realizar un análisis detallado de la situación, gracias a que se cuenta con experiencias de los casos resueltos anteriormente o intentos fallidos en resolver un problema similar. Estos pueden recordar o no en detalles un análisis realizado a una situación anterior, pero pueden reconocer el enfoque dado a esta situación. Una de las ventajas que brinda la construcción de un SBC es que no se pierde el conocimiento del experto, ya que este puede escasear y necesitarse en muchos lugares, además estos sistemas se pueden utilizar para mejorar la calidad del conocimiento de los expertos humanos.

1.4.1.1 Tipos de Sistemas Basados en el Conocimiento

El conocimiento almacenado en la BC puede ser de diferentes tipos: simbólicos, pesos de una red neuronal, ejemplos o casos de problemas del dominio, entre otros. Esto da lugar a la existencia de diferentes SBC, entre los que se encuentran:

- **Sistemas Basados en Reglas**

Los Sistemas Basados en Reglas (SBR) se caracterizan porque la forma de representación del conocimiento son las reglas de producción, y como método de inferencia utilizan la regla de modus ponens (3). Este tipo de regla expresa siempre una condicional, con antecedentes y un consecuente. La interpretación de una regla surge del hecho que si los antecedentes se satisfacen entonces se logra el consecuente. Como ventaja fundamental muestran su capacidad de interpretación y explicación de la inferencia. El proceso de solución de problemas en un SBR es crear una cadena de inferencias que constituye un camino entre la definición del problema y su solución, mediante la utilización del encadenamiento hacia atrás y hacia adelante. Una de las facilidades que brinda es la manipulación de incertidumbre. Entre sus desventajas se encuentra el encadenamiento infinito, este se produce debido a que la mayoría de los SBR realizan una búsqueda primero a profundidad; este método padece dicho problema, si no es implementado correctamente. La adición de nuevo conocimiento puede resultar contradictorio, ya que puede darse el caso que al incorporar nuevas reglas sus conclusiones entren en contradicción con las de las reglas ya existentes en la BC. La modificación de reglas existentes, para considerar casos específicos, puede llevar a un incremento sustancial de la BC si las modificaciones no se realizan convenientemente. El reconocimiento de qué reglas son aplicables en cada ciclo es altamente ineficiente; ya que durante cada ciclo es necesario examinar cada regla para encontrar la aplicable. Hay dominios en que las entradas varían mucho y requerirán miles de reglas para considerar todas las situaciones. Por lo planteado anteriormente se descarta esta técnica para dar solución a la problemática.

- **Sistemas Basados en Probabilidades**

En los Sistemas Basados en Probabilidades la adquisición del conocimiento consiste en coleccionar muestras y realizar un procesamiento estadístico que produzca las probabilidades o frecuencias que forman la base de conocimiento. Éstos utilizan generalmente el Teorema de Bayes como método de solución de problemas (3). A pesar de que los sistemas probabilísticos se consideran uno de los tipos más importantes de sistemas basados en el conocimiento no son factibles para todo tipo de dominio, pues se dificulta construir las redes con ayuda de expertos humanos cuando existen carencias de conocimiento. No son viables para explicar el razonamiento, ya que los métodos y modelos que utiliza están aún lejos de ofrecer explicaciones comprensibles. Este método se utiliza principalmente en sistemas donde la naturaleza de las inferencias sea probabilística. También se utiliza en sistemas

donde el conocimiento provenga de múltiples fuentes, algunas fiables y otras no. Por lo planteado anteriormente no resulta conveniente utilizar esta técnica, ya que no se ajusta a los objetivos que se pretenden alcanzar con la investigación.

- **Sistemas Basados en Frames (SBF)**

  Un frame es una estructura de datos compleja que contiene un agregado de información acerca de un objeto, ofreciendo una representación estructurada de un objeto o una clase de objetos. Uno de los principales mecanismos de inferencia es la herencia. A partir de los frames individuales se crea una organización (taxonomía) de los frames que permite al diseñador de la Base de Conocimientos describir cada clase como una especialización (subclase) de otra más genérica (más abstracta) (3). Los sistemas basados en frames constituyen esencialmente sistemas contestadores a preguntas, pues sus mecanismos de razonamiento resultan débiles; ellos pueden ser considerados SBC rudimentarios. Los mismos presentan algunos inconvenientes que hacen que se descarte esta técnica, como por ejemplo que en muchas situaciones del mundo real se involucran objetos que se diferencian de los prototipos. Lo que trae consigo el incremento de la complejidad del sistema ya que cada instancia particular de un objeto tiene rasgos únicos que tienen que ser representados. Además, cuando aparecen nuevos objetos el SBF no tiene prototipos construidos para guiar la representación de estas instancias. Los frames deben ser usados donde sea necesario tener descripciones estructurales complejas para describir adecuadamente el dominio de la aplicación, pues un frame ofrece una representación estructurada de un objeto o una clase de objetos. Por lo planteado anteriormente se descarta esta técnica para dar solución a la problemática.

- **Redes Expertas**

  En las Redes Expertas la adquisición del conocimiento incluye la selección de los ejemplos, el diseño de su topología y el entrenamiento de la red para hallar el conjunto de pesos. Facilitan el trabajo con información incompleta y brindan algoritmos poderosos de aprendizaje para crear la base de conocimiento. Estas generalmente utilizan pesos como forma de representación del conocimiento y el cálculo de niveles de activación de las neuronas como método de solución de problemas (3). Estos sistemas requieren de muchos ejemplos y son cajas negras que no explican cómo se alcanza la

solución, lo que se considera una gran desventaja para la investigación, ya que se requiere de un mecanismo que muestre como se llegó a la solución propuesta.

- **Sistemas de Razonamiento Basados en Casos**

  En los Sistemas Basados en Casos la adquisición del conocimiento se reduce a la selección de un conjunto de ejemplos o casos resueltos y su organización en la base de casos. Argumenta una solución mediante los casos que son relevantes al nuevo problema. Cada caso es la experiencia anterior almacenada. Su dificultad radica en la definición adecuada de la función de semejanza, al no existir una función de semejanza general apropiada para cualquier problema. En este paradigma la base del comportamiento inteligente de un sistema radica en recordar situaciones similares existentes en el pasado. Los dominios abordados mediante técnicas RBC implican generalmente tareas de análisis, clasificación, interpretación, diagnóstico, diseño, planificación o asesoría (3). Después de haber analizado las diferentes técnicas de la IA para la construcción de sistemas inteligentes, se opta por la utilización del RBC, pues sus modelos de recuperación de casos semejantes resultan adecuados para acceder a información que es relevante a una situación dada, y que puede servir de mucha ayuda en el momento de tomar una decisión. Por otra parte, el grado de veracidad de la solución encontrada se obtiene directamente a partir del grado de semejanza entre el problema y el caso recuperado. Otra diferencia importante con otros planteamientos de IA radica en que el RBC incorpora una concepción incremental y continua del aprendizaje, puesto que cada vez que un problema es resuelto se retiene una nueva experiencia que está inmediatamente disponible para la resolución de futuros problemas. De este modo, el aprendizaje tiene lugar de una forma natural, como un subproducto del propio método de resolución aplicado, mediante la actualización continua de la base de casos. La utilización de una base de casos evita tener que derivar de nuevo soluciones ya obtenidas y recordar problemas previos evita seguir caminos equivocados. Además, a diferencia de los métodos clásicos de solución de problemas de la Inteligencia Artificial la búsqueda de la solución a un problema no se inicia a partir de los datos o el objetivo, por lo que el camino se acorta considerablemente. Otra de las causas por las que se utilizará esta técnica es que existe una alta frecuencia de repetición del desarrollo de productos similares en proyectos cuyas características son similares.

1.4.2 Razonamiento Basado en Casos (RBC)

Los sistemas de razonamiento basados en casos (SRBC) son una de las tecnologías actuales para construir sistemas basados en el conocimiento para la toma de decisiones. Estos sistemas utilizan el razonamiento basado en casos (RBC) como método de solución de problemas para resolver nuevas situaciones. Los SRBC, apoyan sus predicciones en ejemplos (casos) que se almacenan en la fase de aprendizaje. Una función de distancia o semejanza determina los casos más semejantes al nuevo problema y las soluciones de los casos recuperados se adaptan para obtener una (16).

El Razonamiento Basado en Casos (RBC) representa un método para resolver problemas no estructurados, en el cual el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos (3). Debe destacarse que es una técnica, en la cual la memoria se sitúa como fundamento de la Inteligencia Artificial y más concretamente de los sistemas basados en el conocimiento. Denota un método en el cual la solución de un nuevo problema se realiza a partir de las soluciones conocidas para un conjunto de problemas previamente resueltos (o no resueltos) del dominio de aplicación.

La idea básica del RBC es recuperar, adaptar y validar las soluciones encontradas en experiencias previas en un intento de relacionarlas con un problema actual. Las experiencias previas están representadas como una biblioteca de casos que reside en memoria (3). Una de las formas más sencillas de dar solución a un nuevo problema es recuperando el caso más similar, y la solución del mismo se adapta al nuevo problema en un intento para resolverlo.

Como se puede apreciar el RBC es básicamente el procesamiento de la información apropiada, recuperada en el momento oportuno. De modo que el problema central es la identificación de la información pertinente cuando se necesite. El RBC tiene diferentes formas de manifestarse, como por ejemplo puede razonar considerando precedentes, usar los casos viejos para explicar nuevas situaciones o usar los casos viejos para criticar nuevas soluciones. El funcionamiento del RBC parte de estos principios y para ello realiza cuatro actividades fundamentales (Figura 1):

1. Recuperación.
2. Reutilización o adaptación.

3. Revisión.
4. Almacenamiento o retención.

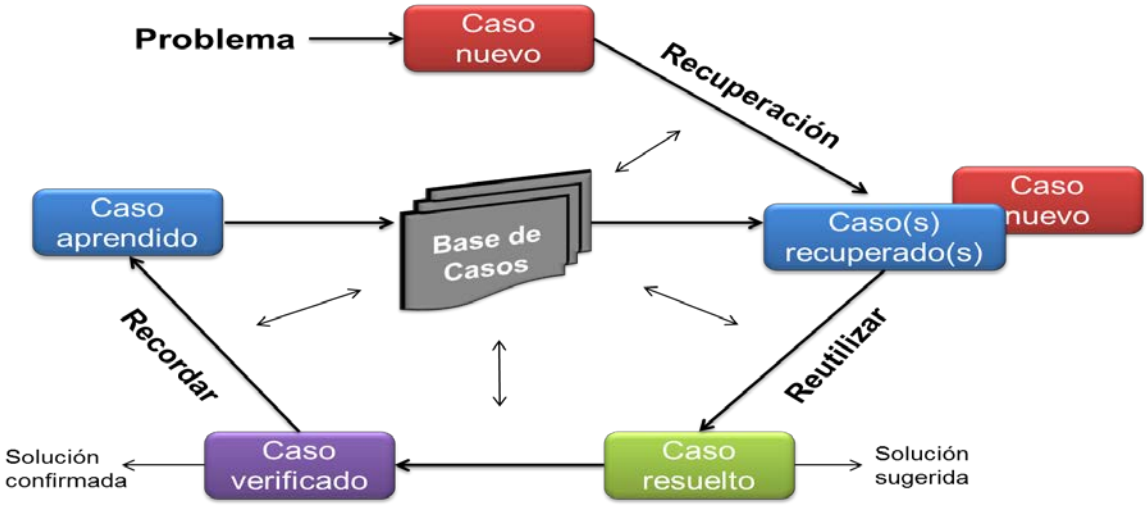


Figura 1 Ciclo de vida de un Sistema Basado en Casos.

Como se muestra en la figura anterior, el RBC consiste en la recuperación de los casos anteriores ya resueltos que sean similares al nuevo caso que se desea resolver; la adaptación de estos al caso a resolver; la revisión de la solución propuesta y finalmente el almacenamiento de la información relevante obtenida.

La elaboración de un sistema que emplea el RBC presenta dos problemas principales: el primero, saber cómo almacenar la experiencia de tal forma que esta pueda ser recuperada en forma adecuada, y el segundo, conseguir utilizar la experiencia previa en un problema actual. La forma de representar y almacenar estas experiencias se realiza a través de casos. Un caso mantiene todos los atributos y características relevantes de un evento pasado. De acuerdo con la naturaleza del problema tratado se define la representación del caso, es decir, cuáles son los atributos importantes, qué problemas serán tratados, cuál es la solución propuesta, etc. Además, es necesario definir el o los mecanismos de recuperación de casos.

Como se puede apreciar los Sistemas Basados en Casos son un tipo especial de SBC. Estos utilizan como base de conocimiento una base de casos, que contiene ejemplos del dominio de aplicación; y como

método de solución de problemas el Razonamiento Basado en Casos (RBC), que actúa directamente sobre los ejemplos o casos almacenados en la BC.

1.4.3 Ventajas y desventajas del razonamiento basado en casos.

El RBC como método de solución de problemas para el desarrollo de sistemas basados en el conocimiento presenta diversas ventajas y desventajas entre las que se encuentran:

Ventajas (3)**:**

- Experiencias previas que hayan sido exitosas pueden ser utilizadas para justificar nuevas soluciones.
- Experiencias previas que no hayan sido exitosas se pueden utilizar para prever problemas.
- La comunicación entre el sistema y los expertos se realiza basándose en ejemplos concretos, es decir, el sistema explica sus decisiones citando precedentes.
- El RBC permite proponer soluciones para los problemas rápidamente, evitando el tiempo necesario para derivar respuestas desde el estado inicial de un proceso de búsqueda de soluciones.
- El RBC permite proponer soluciones en dominios que no se comprenden completamente.
- El RBC es aplicable a un amplio rango de problemas.

Desventajas (3)**:**

- Requiere de una base de datos considerablemente grande y bien seleccionada.
- La consistencia entre varios casos es difícil de mantener.
- El RBC depende de una adecuada función de semejanza la cual no es fácil de encontrar para cada aplicación.

1.5 La Base de Casos

El componente principal de un SRBC, es la base de casos, la cual constituye la memoria del sistema. La BC es el módulo encargado de almacenar y organizar todos los casos disponibles y su estructura es crucial para la fase de recuperación de casos similares. Un caso está definido como "un conjunto de características, atributos, relaciones y resultados" (17); dicho conjunto hace referencia a una situación

específica. Es decir, un caso está formado por varios atributos que dan una descripción del problema y una solución para el caso (18).

El conjunto de problemas resueltos (casos) forman la BC o memoria permanente del sistema, a partir de los cuales el sistema hace sus inferencias. En la BC se registran los casos resueltos; esta debe poseer una organización que le permita mostrar cualidades similares a la memoria humana, es decir, debe ser ilimitada, en la medida que la memoria crezca no se puede hacer más lenta y debe permitir buscar directamente los elementos de memoria que sean relevantes a un problema.

Estructura de los casos

Según Koloder un caso es una pieza contextualizada de conocimiento que representa una experiencia que enseña una lección fundamental para alcanzar los objetivos del razonador (19). Por ese motivo una vez que se adquiere el conocimiento, es necesario encontrar una representación simbólica, clara, precisa y completa del mismo. Las formas de representar el conocimiento son:

- ✓ Atributo-valor: los casos se representan como dos conjuntos no estructurados de pares atributo-valor que representan el problema y las características de la solución. En esta representación cada dato se representa con un número fijo de atributos, junto con la valoración de esos atributos para ese dato. Es decir, los atributos representan las características del problema. Se puede plantear que es el esquema de representación más sencillo.
- ✓ Estructurado: Grafos, redes semánticas, árboles de decisión, etc.

Formas de organización de la Base Casos

Dado que los casos constituyen el elemento principal de todo SRBC, la manera de almacenarlos repercutirá directamente en el rendimiento del mismo. Tradicionalmente se han propuesto dos estructuras principales para el almacenamiento de casos:

Plana: En esta estructura se presentan los casos completos de forma secuencial en una lista simple, un arreglo o un fichero. Esta estructura tiene el principal inconveniente de que la búsqueda de casos es menos eficiente, por lo que no es recomendable utilizarla en sistemas que trabajan con una base de casos

de gran tamaño y necesitan respuesta en tiempo real. Por otra parte, la inserción de nuevos casos en este tipo de estructuras es muy sencilla ya que basta con incluir un nuevo registro con el nuevo (20).

Jerárquica: En una estructura con memoria jerárquica se utilizan representaciones en forma de árbol, en los que cada nodo interior representa un atributo del caso y en las hojas se almacenan las soluciones de los mismos. Cada recorrido desde la raíz hasta una de las hojas del árbol representa un caso completo. La gran ventaja de este tipo de almacenamiento es la eficiencia en la búsqueda, pero a cambio se sacrifica la sencillez de inserción de nuevos casos (20).

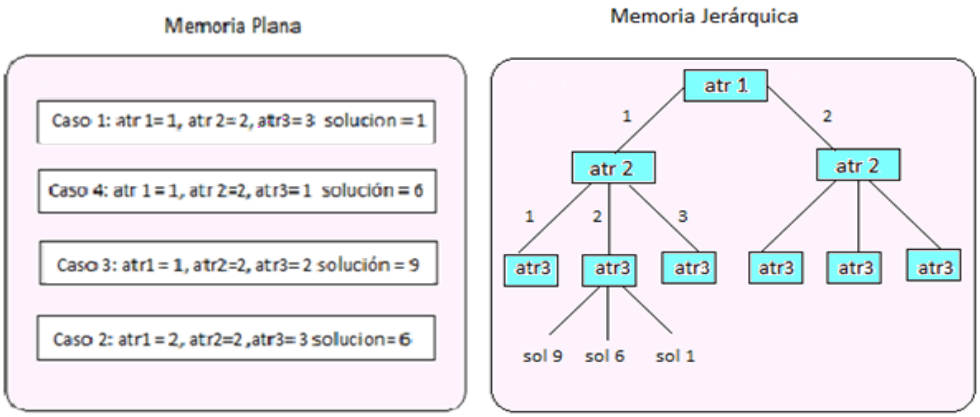


Figura 2 Estructura plana y jerárquica.

1.6 Descripción de los módulos de un SRBC

Los componentes del sistema no se pueden analizar separadamente pues la arquitectura de cada uno puede influir en la eficiencia del trabajo de la otra; por ejemplo, el modelo de la base de casos influye en el modelo del recuperador de casos. A continuación se hace una breve descripción de cada uno de los componentes de un SRBC.

1.6.1 Módulo de Recuperación

La peculiaridad del RBC implica que la solución de un problema se realice recuperando, desde la memoria, los problemas ya resueltos almacenados en la BC que tienen relación con este. Este proceso está compuesto por dos etapas: el acceso a un conjunto de elementos de memoria o casos que resulten

promisorios para el problema a resolver y la recuperación del o los elementos de memoria de este conjunto que sean más semejantes al problema.

La recuperación o selección de los ítems de la memoria permanente, semejantes al problema actual se realiza utilizando una función de semejanza, la cual determina una medida numérica del grado de similitud de cada ítem de la memoria con respecto al problema a resolver. Esta función considera las diferencias y semejanzas entre la descripción de ambos problemas. El mecanismo de recuperación en un SBC asegura que los casos más relevantes sean recuperados para el problema dado, siendo este el resultado más importante del RBC. Los casos relevantes son aquellos que coinciden con la mayoría de los rasgos del problema presente.

En el proceso de recuperación influyen otros elementos que cuentan con gran importancia. Uno de ellos es el tratamiento de la incertidumbre, la incertidumbre se entiende como falta de seguridad o certeza sobre algo. La incertidumbre puede manifestarse de diversas formas y ser provocada por diferentes causas (21). La incertidumbre puede afectar los campos de acción y de decisión o bien perturbar la creencia, fe o validez de un determinado conocimiento. Por eso, los sistemas de información tienen que ser capaces de considerar la incertidumbre. Esta incertidumbre viene dada por la veracidad con que se puede afirmar que un rasgo se comporte de cierta forma. Otro elemento es el umbral de semejanza que no es más que el valor mínimo que debe tener un caso para ser considerado a la hora de dar solución a nuevos problemas.

1.6.2 Módulo de Adaptación

Después de seleccionados y recuperados los casos más parecidos, se procede a elaborar el nuevo, modificándolo de ser necesario. Este nuevo caso obtenido como resultado final será incorporado a la BC como uno más, lo que permitirá que el sistema se desarrolle “aprendiendo” de sí mismo. En la implementación de los métodos de adaptación se hace imprescindible el conocimiento de los expertos en este dominio.

El método de adaptación más simple es no realizar adaptación y transferir la solución vieja al nuevo problema directamente. Pero la adaptación de un caso puede realizarse, ya sea por la transformación de

un caso para ajustarlo a los requisitos de la nueva situación, o mediante la unión apropiada de partes de varios casos. Entre los métodos que se utilizan para realizar la adaptación se encuentran los siguientes:

- Reinstanciación*:* Este método es aplicable cuando se presenta el problema de que los valores de los atributos en el problema actual son distintos de los respectivos valores en el caso previo. Por lo que se debe abstraer el marco del caso previo y su solución, calcular la correspondencia entre los atributos de los dos problemas e instanciar el marco del caso previo y su solución basándose en esas correspondencias.
- Ajuste de parámetros: tipo de técnica de adaptación estructural la cual se utiliza previa o posteriormente a un proceso de reinstanciación. Se tiene el problema de que los parámetros en el problema actual son diferentes de los parámetros en el caso previo, el método consiste en extraer las diferencias y para cada diferencia aplicar una técnica de ajuste especializada. Es decir, se ajustan los parámetros de la solución de casos anteriores de acuerdo con las diferencias entre las descripciones de los casos en cuestión.
- Búsqueda local: es apropiada cuando algún elemento de la solución no corresponde a las necesidades del problema actual. El método consiste en sustituir algún valor buscando un valor relativamente cercano en la jerarquía de abstracciones. Es decir, se realizan búsquedas dentro de jerarquías semánticas y se soluciona el problema por analogía.
- Aplicación derivacional: Como su nombre lo indica es un método derivacional en el cual cuando algún elemento de la solución no corresponde a las necesidades del problema actual se accede al procedimiento mediante el cual se calculó en el caso previo y se reaplica el método.
- Adaptación basada en crítica: Es una técnica estructural en la cual un crítico revisa si una combinación particular de rasgos puede causar un problema. Si se encuentra se aplica una estrategia específica de reparación.
- Sistemas cooperativos: El sistema solo presenta las experiencias relevantes al usuario y este es quien realiza la adaptación. La solución al nuevo problema es resultante de un proceso de cooperación, donde la máquina es la encargada de almacenar y recuperar los casos y la adaptación la realiza el experto humano.

Finalmente, resolver problemas adaptando soluciones conocidas permite al resolvedor de problemas no tener que considerar muchas restricciones, pues solamente hay que tener en cuenta aquellas que se relacionan con los elementos a adaptar.

1.6.3 Módulo de revisión

Una de las características distintivas del razonador basado en casos es la habilidad para aprender de sus experiencias. Para ello el razonador requiere retroalimentación, es decir, conocer qué soluciones dadas por él han sido correctas y cuales erróneas. Sin la retroalimentación, el razonador puede ser más rápido al resolver sus problemas, pero puede repetir sus errores y nunca incrementaría sus capacidades. La evaluación de la solución y su consecuente reparación son importantes contribuciones a la experticia de un razonador basado en casos. La evaluación puede hacerse en el contexto de otros casos similares, basarse en la retroalimentación o en la simulación.

Se puede resumir que la calidad de las soluciones del razonador basado en casos depende de las experiencias que se tienen, de la habilidad para entender nuevas situaciones partiendo de viejas experiencias, y de la destreza en la adaptación y evaluación de soluciones.

1.6.4 Módulo de almacenamiento

En la fase de almacenamiento se procede a almacenar el caso recién resuelto en la base de casos para uso futuro. Este proceso comprende tres fases: determinar qué se necesita almacenar, definir cómo será indexado el nuevo caso y finalmente integrar el caso a la memoria de casos. A este módulo se le conoce también como módulo de aprendizaje.

1.7 Herramientas que utilizan Razonamiento Basado en Casos

A nivel nacional existen herramientas como:

- **Sistema Inteligente de Selección de Información (SISI):** Sistema híbrido que combina Redes Neuronales Artificiales y RBC. Desarrollado en la Universidad Central de Las Villas en el año 1996. SISI es un SRBC del tipo interpretativo en tareas de diagnóstico. El principal problema que presenta SISI es que no es multiplataforma, el programa se ejecuta sobre el sistema operativo Microsoft Windows, por lo que no se puede utilizar en otros sistemas operativos (22).

- Herramienta para la elaboración de sistemas de enseñanza inteligentes (**HESEI**): creada por el grupo de Informática Educativa e Inteligencia Artificial de la Universidad Central de Las Villas (UCLV). Facilita la elaboración de SBC para el proceso de enseñanza-aprendizaje. El trabajo con HESEI se ha extendido a diferentes centros de educación e investigación desarrollándose sistemas en áreas como: Humedales en el Laboratorio de Propagación Masiva de Plantas del Instituto de Biotecnología de las Plantas, UCLV, Teoría de Grafos, Análisis y Diseño de Sistemas y Estructura de Datos (Facultad de Matemática Física y Computación, UCLV) (16).

A nivel internacional:

- **CHEF:** Es un planificador basado en casos, cuyo dominio son las recetas de cocina. CHEF crea las nuevas recetas a partir de otras ya conocidas, como respuesta a unos requisitos (submetas) de platos con ingredientes específicos, sabores particulares. Este sistema tiene que construir planes que satisfagan un número dado de metas simultáneamente (23).
- **PROTOS**: Implementa clasificación y adquisición de conocimiento basado en casos, dada una descripción de una situación u objeto la clasifica por su tipo. Cuando clasifica un elemento de forma incorrecta, su consultor experto le informa del error y del conocimiento que debe usar para clasificarlo correctamente. El dominio de PROTOS es el de los desórdenes auditivos (23).
- **SHYSTER**: sistema jurídico basado en casos. Utiliza técnicas estadísticas para cuantificar la similitud entre los casos, y elige los casos sobre la base de esa medida de similitud (24).

Ninguna de estas aplicaciones es factible para dar solución al problema planteado, los mismos no se adaptan a las características del dominio del problema, no incluyen o tratan los procesos relacionados con la clasificación de los proyectos de desarrollo de software. Además de que no se pueden tener en cuenta para la confección de la BC, ya que esto depende del tipo de información que va a almacenar en la misma.

1.8 Tecnologías y Herramientas

1.8.1 Metodología de desarrollo de software

Para asegurar el éxito durante el desarrollo de software no es suficiente contar con notaciones de modelado y herramientas, hace falta un elemento importante: la metodología de desarrollo, la cual provee

una dirección a seguir para la correcta aplicación de los demás elementos. Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software, indicando quién debe hacer cada actividad, cuándo hacerla y qué debe hacer (25). Actualmente las metodologías se clasifican en Metodologías pesadas y Metodologías ágiles.

Las metodologías ágiles, tienen como común denominador un modelo de desarrollo incremental para producir tempranamente pequeñas entregas en ciclos rápidos, y predisposición para el cambio y la adaptación continua; según sea la conformidad o no de lo producido, y las modificaciones propuestas por los usuarios. Estas metodologías por lo general se centran en desarrollar productos funcionales más que en conseguir una buena documentación (26). Son metodologías centradas en la implementación, que evita cualquier tipo de documentación fuera del código. Por otra parte, las Metodologías Pesadas están orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Teniendo en cuenta lo antes expuesto y la necesidad de contar con la documentación necesaria para la futura implementación del sistema, se propone el uso de la metodología pesada Proceso Unificado de Rational (RUP).

Proceso unificado de Rational / Rational Unified Process (RUP).

RUP es una metodología de desarrollo de software que intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos de software. Las características principales están dadas en que es un proceso iterativo e incremental, centrado en la arquitectura y dirigido por casos de usos. La relación entre estas tres características es la siguiente: “La arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración” (27).

Para el desarrollo de este trabajo se seleccionó la Metodología de Desarrollo de Software RUP, pues genera con su utilización gran cantidad de documentación, cuestión importante en la investigación, ya que facilitará el trabajo futuro de los desarrolladores o administradores. Además, con RUP se hace una adecuada captura de requisitos, lo cual sin dudas contribuirá a que el trabajo sea más rápido y que a la vez se cumpla con las expectativas y funcionalidades que se quieren lograr. Brinda también la facilidad de que una vez detectado un posible error se pueda retroceder y corregirlo, sin que atente en un futuro con el correcto funcionamiento del sistema. Se ajusta a cualquier envergadura de proyectos y equipos de trabajo.

Tiene gran cantidad de roles bien definidos, que actuarán en cada una de las fases y flujos de trabajo. Cuenta con variedad de diagramas para el modelado en las distintas fases, lo cual hace más entendible el proceso, para cualquier interesado.

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para preparar todos los esquemas de un sistema software. De hecho, UML es una parte esencial del Proceso Unificado (27).

1.8.2 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener, y controlar la información sobre tales sistemas (28). Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. UML ayuda al usuario a entender la tecnología en realidad y la posibilidad de mostrar una idea antes de invertir en proyectos que no estén seguros en su desarrollo. Entre las propiedades que tiene este importante lenguaje se encuentra que modela estructuras complejas, soporta estructuras orientadas a objetos, como clases, componentes y nodos.

1.8.3 Herramienta Case

Como herramienta Case se propone la utilización del Visual Paradigm. La misma es una herramienta que es soportada por cualquier sistema operativo. Es una herramienta visual de ingeniería de software para el modelado, que tiene un entorno de trabajo que muestra colección de menús, barra de herramientas y ventanas que permite realizar diagramas. Puede ser utilizada por una gran variedad de usuarios como analistas de sistemas, analistas de negocios e ingenieros de software. Provee soporte para la generación de código y la ingeniería inversa. Se integra con algunas herramientas como: Eclipse, Netbeans, Jbuilder, Oracle, entre otras. Se utilizará Visual Paradigm Suite versión 6.4.

1.8.4 Lenguajes de programación

Los lenguajes de programación permiten la creación de programas. Estos facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por las personas, a su vez resultan independientes del modelo de computador a utilizar. Para la futura implementación del sistema se propone el uso de Groovy.

Groovy

Groovy es un lenguaje de programación con una sintaxis análoga a Java y que corre sobre la Java Virtual Machine (JVM). Se integra completamente con Java, le permite mezclar y acoplar al código Groovy y Java con un esfuerzo mínimo (29). Groovy es un lenguaje dinámico, todo ocurre en tiempo de ejecución, incluida la lógica de gestión de llamadas a métodos y acceso a propiedades. A diferencia de otros lenguajes que se adaptaron para la JVM, Groovy fue diseñado para la JVM, por lo que no existe incompatibilidad (30).

La sintaxis de Groovy es más flexible y poderosa que la de Java. Las docenas de líneas de código en Java pueden acortarse a pocas líneas de código en Groovy ganando en legibilidad, mantenibilidad y eficiencia. Entre las características que distinguen a Groovy incluyen el tipado estático y dinámico, closures, sobrecarga de operadores, sintaxis nativa para listas, expresiones regulares, expresiones embebidas dentro de strings. El framework que utiliza este poderoso lenguaje, basado y construido para la madura y robusta tecnología Java es Grails. Se propone que se use la versión de Groovy 1.8.5.

1.8.5 Framework

La palabra inglesa framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Grails

Es un framework de desarrollo web dinámico para la plataforma Java. Utiliza la flexibilidad de Groovy para proporcionar un dominio específico del lenguaje (DSL) para el desarrollo web. El objetivo es desarrollar aplicaciones web con el mínimo esfuerzo, sin tener que repetirse (31). Las aplicaciones desarrolladas con este framework utilizan el patrón MVC (Modelo-Vista-Controlador). Grails está construido sobre sólidos proyectos de código abierto, como Spring, SiteMesh, GORM/Hibernate. Se puede usar Grails como un entorno de desarrollo independiente que esconde todos los detalles de la configuración o integra la lógica del negocio hecha en Java. Grails apunta a hacer el desarrollo tan simple como sea posible. No centra su desarrollo en reinventar lo bueno, sino mejorarlo: trata de tomar lo mejor de las tecnologías. Como ejemplo se puede citar que Grails crea un simple DSL, que simplifica el trabajo con Hibernate, dejando fuera las complicadas configuraciones de mapeo en archivos XML.

La integración de Groovy y Grails permite a los desarrolladores web la construcción de sistemas robustos, bajo un entorno coherente y fiable, con un mínimo de esfuerzo. Donde se emplean conceptos como la reutilización de código, la metaprogramación, DSL, además soporta Ajax y es multiplataforma. Cuenta con una gran comunidad de desarrolladores que lo mantiene en constante avance. Además, una de las potencialidades de Grails es su bajo costo de instalación ya que solo necesita que estén instalados dos elementos: la JVM y el framework Grails 1.1 o versiones superiores. Se propone que se use la versión de Grails 2.0.0.

1.8.6 Gestor de Base de Datos

Los Sistemas Gestores de Bases de Datos (SGBD), son software que permiten la administración y mantenimiento de los ficheros y datos de una Base de Datos o de varias. Proporcionan funcionalidad añadida al sistema de ficheros para facilitar la gestión de datos. Además, proporcionan valor añadido, como seguridad en cuanto a acceso, copias de respaldo, entre otras. En la actualidad existen disímiles SGBD. Para la futura implementación del sistema se propone el uso de PostgreSQL, esencialmente porque es de software libre, posibilitando su uso sin restricciones.

PostgreSQL

Dentro de los gestores de bases de datos existentes, se nombra como uno de los más distintivos. Está diseñado para soportar volúmenes masivos de datos, sin que ello afecte en lo absoluto su rendimiento.

Ofrece una fortaleza adicional sustancial al incorporar cuatro conceptos adicionales básicos: Clases, Herencia, Tipos, Funciones. Cuenta además con características que aportan potencia y flexibilidad adicional: Restricciones, Disparadores, Reglas, Integridad transaccional (32).

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. Posee soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Corre en la casi totalidad de los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc. Soporta el protocolo de comunicación encriptado por SSL. Posee utilidades para limpieza de la base de datos y para el análisis y optimización de Query's. Se propone el uso de PostgreSQL 8.4.

Conclusiones parciales

En este capítulo se realizó un estudio de los diferentes criterios de clasificación del software y se propuso la utilización del criterio "producto final" que se genera en el proyecto. Se analizaron las técnicas de inteligencia artificial usadas para la construcción de sistemas inteligentes basados en el conocimiento, llegando a la conclusión de que el razonamiento basado en casos es el que más se adapta para dar solución a la problemática planteada, ya que los proyectos pueden ser representados como casos. Se seleccionó RUP como metodología de desarrollo de software; UML como lenguaje de modelado y Visual Paradigm for UML 6.4 para la realización de los diagramas. Se propuso Groovy 1.8.5 como lenguaje de programación en conjunto con el framework Grails 2.0.0, así como el gestor de base de datos PostgreSQL 8.4 para la futura implementación de la aplicación web.

CAPÍTULO II: DISEÑO DE LA BASE DE CASOS

Introducción

En el presente capítulo se realiza el proceso de adquisición del conocimiento. Se identifican los rasgos predictores y objetivos que conformarán los casos. Posteriormente se define la estructura de los casos así como su organización en la base de casos. Se proponen los algoritmos a utilizar en la fase de recuperación. Finalmente, se valida la propuesta de solución.

2.1 Adquisición del conocimiento

La adquisición del conocimiento es la extracción del conocimiento de los expertos humanos, libros, documentos, archivos de computadora, entre otros. El proceso de la adquisición del conocimiento, con frecuencia es el componente más difícil, por ello es común que se presente como el mayor impedimento o cuello de botella en el desarrollo de un sistema experto. Resulta indispensable la estructuración e implementación del conocimiento del experto humano lo cual implica, una gran cantidad de trabajo así como el establecimiento de comunicaciones ampliadas entre el experto humano y el ingeniero de conocimiento, enfrentando los problemas asociados con esta actividad.

2.1.1 Métodos para la adquisición del conocimiento

Para la identificación los rasgos predictores y objetivos se seleccionaron métodos para la adquisición del conocimiento tales como:

- **Conocimiento documentado**: desde el inicio de la investigación se comenzó a extraer el conocimiento de diversas fuentes implicadas (libros, artículos de revistas, manuales, informes y documentos de ayuda), en formato digital o impreso, ya que cualquier documento relacionado con la clasificación de los proyectos informáticos y sus características podía ser útil para la confección de la base de casos.
- **Entrevistas no estructuradas y semiestructuradas**: se detectó que el nivel de experiencia en el dominio del conocimiento de la mayoría de los expertos humanos era alto, por lo que se realizaron

entrevistas a los mismos que permitieran al ingeniero del conocimiento detectar la información importante del dominio a tratar.

- **Observación directa**: a través de la observación del proceso de desarrollo del software en los proyectos, se logró comprobar la coherencia con la información obtenida de los documentos y las entrevistas.

2.1.2 Selección del experto

Para que el proceso de adquisición del conocimiento tuviera éxito, fue imperativo que expertos humanos (EH) estuvieran disponibles y que contaran con las siguientes características:

- Los expertos deben tener la disposición a dar conocimiento y colaborar en los esfuerzos de desarrollo.
- Los expertos existentes deben estar posibilitados para resolver problemas en el dominio del tema.
- Los expertos deben articular razonablemente el conocimiento. Deben ser capaces de describir el conocimiento del dominio y cómo se debe aplicar.
- Los expertos deben estar vinculados a los procesos de producción y la gestión del software en la universidad.
- Los expertos deben tener experiencia en el trabajo en proyectos.

Después de analizados los indicadores quedaron seleccionados los expertos que se muestran en el Anexo 1. Los mismos son expertos con más de dos años de experiencia, entre los que se encuentran especialistas de la DGP, una profesora con maestría en gestión de proyectos e ingenieros informáticos con gran experiencia en el desarrollo de proyectos informáticos.

2.2 Selección de los rasgos objetivos

En el Capítulo I se propuso realizar la clasificación de acuerdo al producto final, ya que esto permitirá a sus responsables encontrar escenarios similares. Por una parte, será información clave para quienes toman las decisiones, debido a la facilidad que brinda la agrupación de los proyectos y por otra, se puede convertir en una interesante estrategia para los implicados, permitiéndoles describir su proyecto, la identificación de errores y desarrollar un producto de calidad. Esto proporcionará resultados más reales para la estimación del tiempo de duración de los proyectos.

Se identificó una gran variedad de software desarrollados en los proyectos para dar solución a disímiles problemas, como por ejemplo los sistemas de gestión, aplicaciones de realidad virtual, aplicaciones de multimedia, sistemas empotrados, sistemas criptográficos, entre otros. Para la propuesta de solución solo se tendrán en cuenta los primeros 4 mencionados anteriormente, ya que se encuentran entre los más representativos en la Universidad.

Descripción de los productos de software a tener en cuenta en la solución

Sistemas de Gestión

Los sistemas de gestión se encargan esencialmente del manejo o gestión de la información referente a cualquier proceso. La gestión de la información se puede definir como el conjunto de actividades realizadas con el fin de controlar, almacenar y posteriormente recuperar adecuadamente la información producida, recibida o retenida por cualquier organización en el desarrollo de sus actividades. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en puntos de ventas) (33). Estos sistemas generalmente implementan el patrón CRUD.

Aplicación Multimedia

Multimedia es cualquier combinación de texto, arte gráfico, sonido, animación y video que llega a las personas por computadora u otros medios electrónicos. Es un tema presentado con lujos de detalles (8). Este tipo de aplicaciones se emplea para disímiles tareas como son el entretenimiento, la enseñanza/aprendizaje, el marketing, entre otras. Esencialmente se concentra en llamar la atención del usuario.

Realidad Virtual

La Realidad Virtual consiste en simular todas las posibles percepciones de una persona, como los gráficos para la vista, sonido, tacto e incluso sensaciones de aceleración o movimiento. Es una simulación tridimensional generada o asistida comúnmente por computadora de algún aspecto del mundo real o ficticio, en el cual el usuario tiene la sensación de pertenecer a ese ambiente sintético o interactuar con él (34).

Sistemas empotrados o embebidos

Programas que se ejecutan dentro de un equipo o hardware, son aplicaciones de propósitos específicos y realizan pocas funcionalidades. Se encuentran limitados con respecto al procesamiento y al uso de memoria, ya que deben ser bajos. Son confeccionados utilizando lenguajes tales como ensamblador, C++, C o Java en última instancia, ya que este necesita muchos recursos. Son aplicaciones que casi siempre dan respuesta en tiempo real. Ejemplos de software empotrados son los que se encuentran en los celulares, cajeros automáticos, semáforos, etc.

2.3 Selección de los rasgos predictores

Después de definidos los valores que pueden tomar los rasgos objetivos es necesario identificar los rasgos predictores. Estos rasgos y sus posibles valores fueron obtenidos a través del análisis del conocimiento documentado y mediante entrevistas a profesores que se encuentran vinculados a la producción. Para las entrevistas se utilizaron las preguntas que se encuentran en el Anexo 2.

Rasgos identificados

La mayoría de los expertos identificaron elementos comunes, lográndose así la identificación de los diferentes rasgos o elementos que influyen en el desarrollo de los distintos tipos de software en los proyectos de la UCI. A continuación se muestra una tabla con los rasgos identificados, su descripción, los posibles valores que pueden tomar y si son de Multiselección o no, es decir, si un rasgo puede tomar más de un valor.

Rasgo	Descripción	Posibles valores	Multiselección
Objeto social	Es básicamente a lo que se dedica o se utiliza el sistema.	Gestión de la información, valor agregado, simulación, diseño, enseñanza, aprendizaje, marketing.	No

Tipo de datos	El tipo de un dato es el conjunto de valores que puede tomar durante el programa. En este caso será el tipo de datos que maneja el sistema.	Ordinarios [1] , tridimensionales, medias, archivos de configuración, binario, ASCII.	Si
Diseño arquitectónico	Facilita la comunicación entre todas las partes interesadas en el desarrollo de un sistema basado en computadora.	Cliente/Servidor, N capas, Orientado a componentes.	Si
Patrones	Los patrones tanto de diseño como arquitectónicos que utilizan para guiar la búsqueda de las soluciones a problemas comunes en el desarrollo de software.	MVC, MVCMM, GOF, GRASP.	Si
Lenguaje de programación	Un lenguaje de programación que se utiliza para la confección de los sistemas.	JavaScript, C, C++, PHP, C#, ensamblador, MikroC, Visual Basic, HTML, Java, Groovy, Ruby, ASP.NET, Delphi, PHYTON.	Si
Hardware	Hardware que necesita cada tipo de sistema para su funcionamiento. Puede que un sistema no necesite	Servidor, tarjeta gráfica, teléfonos, cajeros automáticos, semáforos, switch, no necesita.	Si

[1] Ordinarios: datos lineales, numéricos, texto, entre otros.

	hardware específico.		
Tecnología utilizada	Tecnologías o programas que se utilizan para la confección de los diferentes tipos de sistemas.	CSS, SPRING, JSF, HIBERNATE, SGBD (POSTGRES, ORACLE), SYMFONY, CODE IGNITER, Matlab Simulink, MikroCforPic, ToolBook, Ktoon, Mediator, Macromedia Flash MX, Scala Multimedia MM200, Windows Movie Maker, Adobe Photoshop CS, Xara X1, Flax, Flash, Macromedia Authorware, REND386 v.5, MULTIVERSE, 3D-Studio, 3D Studio Max, Maya, Blender.	Si
Características especiales de seguridad	Muestra cómo se logra la seguridad en cada uno de los sistemas.	Seguridad a nivel de los lenguajes de programación, Gestión de accesos e identidades, Seguridad informática de datos e información, Codificación de la información, no implementan seguridad.	No

Tabla 1 Rajos identificados, descripción, posibles valores y multiselección.

2.4 Representación del conocimiento

Una vez realizada la adquisición del conocimiento, y obtenidos los rasgos predictores (características de los proyectos) y objetivos (clasificaciones de los proyectos) que conformarán un caso, se hace necesario definir las estructuras que tendrán los casos y la BC.

Estructura de los casos

Los casos estarán descritos a partir de los valores que se le asignen a los rasgos predictores y a los rasgos objetivos. Los rasgos predictores son los que determinan los valores de los rasgos objetivos. Para

la representación de los casos se propone el uso una colección de rasgos, que serán representados de la forma par atributo-valor. Esta representación posee gran sencillez de implementación, trayendo consigo la disminución del tiempo a emplear para el desarrollo de la solución. A continuación se muestra un ejemplo representativo de la estructura de un caso.

Caso 1: objeto_social = gestión de información, patrón = MVC ……. solución = Sistema de gestión

Estructura de la memoria

En la creación de la estructura que formará la BC hay que tener en cuenta aspectos fundamentales como: el tamaño que se aspira que tenga la BC, los requerimientos que se imponen en el dominio específico en el que se trabaja y la inserción eficiente de nuevos casos (35). Teniendo en cuenta estos aspectos se seleccionó la estructura de memoria plana para la BC. Con esta estructura se logrará realizar un análisis exhaustivo sobre los casos para recuperar los mejores. A diferencia de las memorias jerárquicas estas no almacenan datos auxiliares para realizar la búsqueda minimizando el espacio que ocupan los casos (3). A continuación se muestra ejemplo representativo de la organización de los casos en la base de casos.

Base de casos “Clasificación de proyectos”
Caso 1: objeto_social = gestión de información, patrón = MVC ……. solución = Sistema de gestión
Caso 2: objeto_social = valor agregado, patrón = MVC ……. solución = Sistema empotrado
Caso 3: objeto_socia l= gestión de información, patrón = MVC ……. solución = Sistema de gestión
Caso 4: objeto_socia l= simulación, patrón = GOF……. ……. solución = Sistema de realidad virtual
Caso 5: objeto_social = gestión de información, patrón = MVC ……. solución = Sistema de gestión

2.4.1 Tratamiento de la incertidumbre

La incertidumbre se definirá para valores en el intervalo de 0 a 1, teniendo en cuenta que mientras mayor sea el valor de incertidumbre, mayor grado de certeza tendrá el rasgo. La misma será proporcionada por

el experto a la hora de la inserción del rasgo. El uso de la memoria plana facilitará la obtención de cada valor de la incertidumbre de los casos de la BC, permitiendo obtener el comportamiento promedio de la incertidumbre de un rasgo. Esto posibilita dar un valor de incertidumbre para un patrón de búsqueda que no defina este valor. Para la obtención de este valor se propone el uso de la ecuación de la media aritmética.

La media aritmética de un rasgo ($\bar{x}_j$) se obtiene mediante la siguiente función (36):

$$\bar{x}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}$$

Donde:

xij: cantidad de ocurrencias del valor del rasgo j en la base de casos.

n: Cantidad de ocurrencias del rasgo.

2.4.2 Funciones de comparación entre rasgos y función de semejanza entre casos

La semejanza entre los rasgos está dada por la relación entre los valores de dominio de estos. A continuación se detallan las funciones de comparación entre rasgos en dependencia de su dominio.

Para los rasgos con dominios nominales se hará uso de la Distancia de Hamming (37), donde el valor será 1 si son iguales los valores del rasgo del problema y del rasgo del caso recuperado, y 0 cuando no sean iguales. Los casos con mayor cantidad de atributos iguales, serán los más semejantes. A continuación se muestra la función:

$$\delta_i(O_0, O_t) = \begin{cases} 1 \rightarrow x_i(O_0) = x_i(O_t) \\ 0 \rightarrow x_i(O_0) \neq x_i(O_t) \end{cases}$$

Para los rasgos con valores conjuntuales se utilizará la Distancia de Jaccard (37). La similitud Jaccard compara dos conjunto de tokens (palabras), en cuanto a la cantidad de tokens que está presente en cada uno por sobre la cantidad total de ellos en ambos conjuntos.

$$D(X,Y) = \frac{(|X \cap Y|)}{(|X \cup Y|)}$$

Donde X e Y representan los conjuntos de palabras antes mencionados.

La función de semejanza determina el grado de proximidad entre los casos. Teniendo en cuenta que es de gran importancia el tratamiento de la incertidumbre se propone el uso del modelo matemático presentado por la Dra. Martínez, Dr. García, Dr. García (38):

$$\beta(O_0, O_t) = \frac{\sum_{i=1}^{n} p_i \, \delta_i \, (O_0, O_t) \, (1 - \, |\mu_i(O_0) \, - \, \mu_i(O_t) \, |)}{\sum_{i=1}^{n} p_i}$$

Donde:

n: Número de rasgos predictores.

p_i: Peso o relevancia del rasgo i.

μi: valor de la incertidumbre del rasgo O.

$\delta_i \, (O_0, O_t)$: Función de comparación entre los casos O_0 y O_t atendiendo al rasgo i.

2.4.3 Umbral de semejanza

El valor del umbral está relacionado con el comportamiento de los casos en su semejanza, un umbral muy bajo puede conllevar a que se tengan muchos casos en la recuperación y un umbral muy elevado puede conllevar a que no se pueda dar una respuesta en la búsqueda. De ahí que el umbral sea dado por un experto. En caso de que el experto tenga problemas con la definición del umbral, el propio módulo de recuperación debe ser capaz de brindar la posibilidad de calcular el valor umbral de los rasgos.

Para ello construye una matriz cuadrada donde la fila y las columnas están representadas por los casos que se encuentran almacenados en la BC y en la intersección está el valor de semejanza (β).

	$Caso_1$	**$Caso_2$**	**$Caso_3$**	**$Caso_n$**
$Caso_1$	$\beta(C_1, C_1)$	$\beta(C_1, C_2)$	$\beta(C_1, C_3)$	$\beta(C_1, C_n)$
$Caso_2$	$\beta(C_2, C_1)$	$\beta(C_2, C_2)$	$\beta(C_2, C_3)$	$\beta(C_2, C_n)$
$Caso_3$	$\beta(C_3, C_1)$	$\beta(C_3, C_2)$	$\beta(C_3, C_3)$	$\beta(C_3, C_n)$
$Caso_n$	$\beta(C_n, C_1)$	$\beta(C_n, C_2)$	$\beta(C_n, C_3)$	$\beta(C_n, C_n)$

Tabla 2 Matriz de semejanza entre casos.

Para permitir que se tengan en cuenta los casos de la BC, la Dra. Natalia Martínez propone (38) un cálculo del umbral de semejanza, que viene dada por una media aritmética con los valores de semejanza entre los casos, mediante la siguiente expresión:

$$\beta_0 = \frac{2}{m\,(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta\,(O_0\,, O_t)$$

Donde:

m: Número de casos

i: Casos en filas

j: Casos en columnas

$\beta\,(O_0\,, O_t)$: Función de semejanza entre los rasgos O_o y O_t.

2.5 Validación de la propuesta de solución. Método Delphi.

Toda nueva propuesta debe pasar un proceso de prueba y perfeccionamiento; los métodos de predicción posibilitan predecir el comportamiento de un evento específico de la propuesta de solución. En la presente investigación se utiliza el método Delphi, basado en el criterio de expertos.

Este método consiste en la selección de un grupo de especialistas a los que se les pregunta su opinión sobre cuestiones referidas a acontecimientos del futuro. La capacidad de predicción del método se basa en la utilización sistemática de un juicio intuitivo emitido por un grupo de especialistas. El método Delphi procede por medio de la interrogación a especialistas con la ayuda de cuestionarios, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos. La encuesta se lleva a cabo de una manera anónima por lo que la calidad de los resultados depende, sobre todo, del cuidado que se ponga en la elaboración del cuestionario y en la elección de los especialistas consultados (39).

A continuación se detallan los pasos necesarios para la aplicación del método, pero antes se deben mencionar los aspectos fundamentales que este presenta:

1. Selección de los especialistas.
2. Elaboración del cuestionario para la validación de la propuesta.
3. Desarrollo práctico y explotación de los resultados.

2.5.1 Selección de los especialistas

En la investigación se considera que un experto en el tema a tratar se refiere a: “Especialistas que son capaces de brindar criterios terminantes sobre el proceso de desarrollo de software y las características que los mismos poseen”.

Teniendo en cuenta lo planteado anteriormente, se realiza la selección de especialistas bajo las siguientes condiciones:

- Debe ser graduado de nivel superior.
- Debe estar vinculado al desarrollo de proyectos informáticos o tener conocimientos en la gestión de proyectos.
- Debe contar con más de tres años de experiencia en el tema.

La selección de expertos atendiendo a estos criterios, proporciona la obtención de resultados con calidad, junto a otras cualidades propias, que pueden ser: la honestidad, la sinceridad y responsabilidad, haciendo que las opiniones brindadas sean confiables y válidas para el objetivo propuesto.

Para determinar cuáles de los candidatos participará en la evaluación de la solución, se calculó el coeficiente de competencia K, haciendo uso de la siguiente fórmula matemática.

$$K = \frac{(K_c - K_a)}{2}$$

Donde:

K_c: es el coeficiente de conocimiento.

K_a: es el coeficiente de argumentación.

Para el cálculo el Kc y Ka se realiza una encuesta (Anexo 3) a los candidatos a especialistas.

Para calcular el Kc se le solicita al posible experto que dé su criterio sobre los conocimientos que posee del tema. Para esto se utiliza un rango de 0 a 10, considerando que 0 es no tener ningún dominio del tema y 10 es tener pleno dominio del tema. Posteriormente este valor obtenido se multiplica por 0.1 para obtener el coeficiente en un rango de 0 a 1. El experto debe marcar con una cruz (X) en la casilla que

estime pertinente. En el Anexo 4 se muestra la tabla con la respuesta dada por cada especialista en cuanto a su nivel de conocimiento en el tema.

Para calcular Ka, el especialista candidato debe marcar según su consideración, cuáles fueron sus fuentes para la obtención del conocimiento que le permite argumentar su evaluación del nivel de conocimiento que especificó anteriormente. El resultado obtenido se encuentra reflejado en el Anexo 4.

Para calcular el coeficiente de argumentación de las respuestas de los especialistas se traducen a puntos según lo que muestra la siguiente tabla patrón, este se calcula sumando los valores de la tabla patrón en concordancia con las respuestas dadas por los especialistas.

Fuentes de argumentación	**Grado de influencia**		
	Alto	Medio	Bajo
Análisis realizado por usted	0,3	0,2	0,1
Experiencia	0,5	0,4	0,2
Trabajos de autores nacionales	0,05	0,05	0,05
Trabajos de autores extranjeros	0,05	0,05	0,05
Su propio conocimiento del tema	0,05	0,05	0,05
Su intuición	0,05	0,05	0,05
Total	1	0,8	0,5

Tabla 3 Tabla patrón para determinar el coeficiente de argumentación.

El código de interpretación de tales coeficientes de competencias es:

- Si 0,8 < k < 1,0 el coeficiente de competencia es Alto.
- Si 0,5 < k < 0,8 el coeficiente de competencia es Medio.
- Si k < 0,5 el coeficiente de competencia es Bajo.

En la siguiente tabla se muestran los resultados obtenidos en la encuesta de autovaloración realizada a los especialistas candidatos.

Especialista	**Kc**	**Ka**	**K**	**Grado**
1	0,8	1	0,9	Alto
2	0,9	1	0,95	Alto
3	0,8	0,9	0,85	Alto
4	0,8	1	0,9	Alto
5	0,9	1	0,95	Alto

6	0,9	0,9	0,9	Alto
7	0,9	1	0,95	Alto

Tabla 4 Resultados obtenidos en el cuestionario realizado.

Después de analizado el coeficiente de competencia de cada uno de los candidatos a especialistas, se obtuvo como resultado que ninguno de ellos tiene su coeficiente de competencia bajo o medio, por lo que los 7 están aptos para conformar el panel de especialistas.

2.5.2 Elaboración del cuestionario para la evaluación de la propuesta.

Para la evaluación de la propuesta de solución se utiliza el cuestionario que se encuentra en el Anexo 5. Entre los objetivos que se persigue con la realización del mismo, se encuentran poder determinar la necesidad de la propuesta para dar solución a la problemática planteada, determinar el nivel de completitud de las clasificaciones y sus rasgos y la consistencia de las estructuras de la base de casos.

En el cuestionario (Ver Anexo 5), primeramente se solicitan los datos personales de los especialistas y posteriormente se originan ocho preguntas, todas de tipo contable y permitiendo además que en cada una de las preguntas los especialistas emitan sus criterios y hagan recomendaciones con el objetivo de mejorar los resultados de la investigación.

Para el posterior análisis de los cuestionarios realizados a los especialistas se tuvieron en cuenta criterios de evaluación cualitativa y a cada uno de estos criterios se les otorgó una puntuación entre 1 y 5 para su posterior análisis, definiéndose la siguiente tabla:

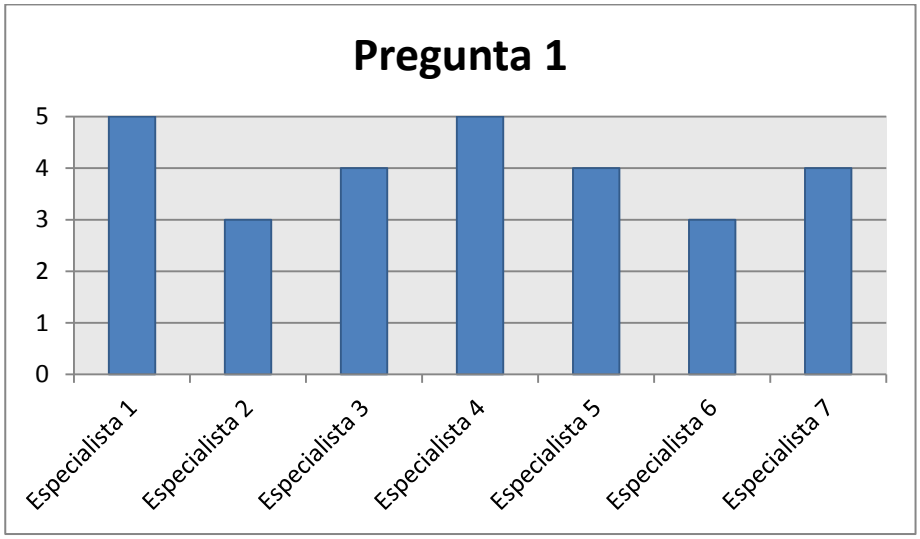
Criterio de Evaluación	Puntuación
Muy útil	5
Bastante útil	4
Útil	3
Poco útil	2
Inútil	1

Tabla 5 Criterios de Evaluaciones.

2.5.3 Desarrollo práctico y explotación de los resultados

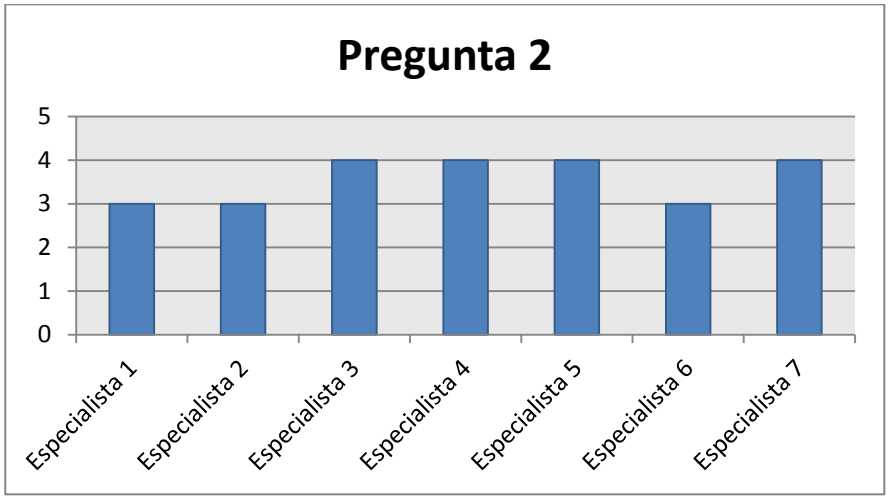
Para ir almacenando los resultados aportados por los especialistas se confeccionaron tablas utilizando el programa “Microsoft Office Excel 2010”. Los gráficos obtenidos por cada una de las preguntas realizadas se muestran a continuación.

En la Gráfica 1 se encuentra la respuesta correspondiente a la pregunta uno de la encuesta, donde los especialistas responden de forma cualitativa según su consideración en qué medida es útil para el cálculo de la estimación de la duración de los proyectos clasificarlos, teniendo en cuenta el tipo de producto final que generan.



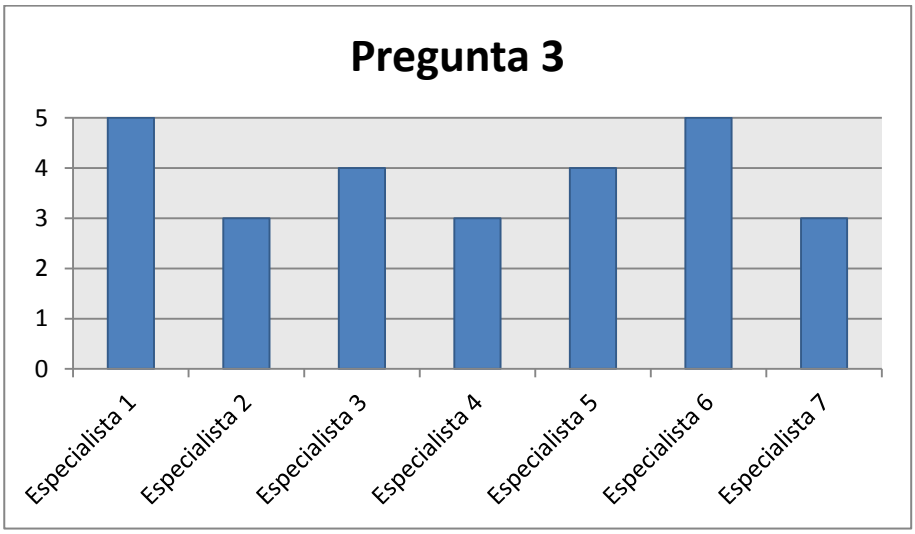
Gráfica 1 Respuesta a la pregunta 1 de la encuesta.

En la Gráfica 2 se encuentra la respuesta correspondiente a la pregunta dos de la encuesta, donde los especialistas responden en qué medida consideran que las clasificaciones propuestas son las adecuadas.



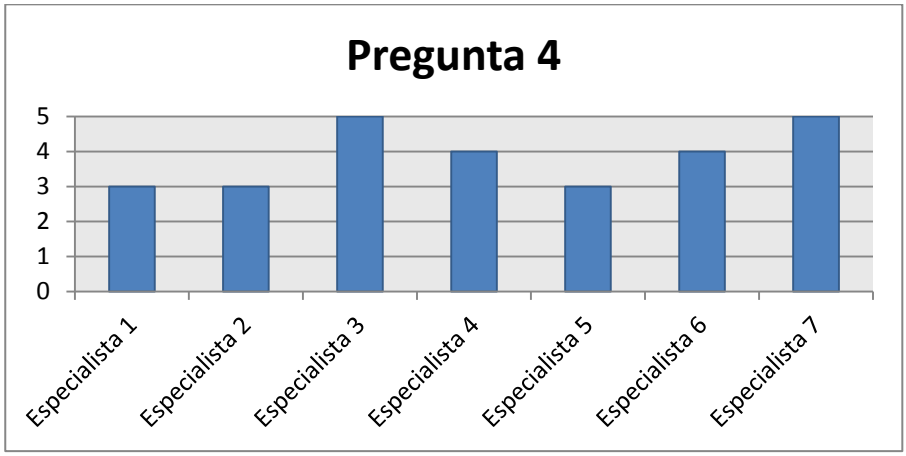
Gráfica 2 Respuesta a la pregunta 2 de la encuesta.

En la Gráfica 3 se encuentra la respuesta correspondiente a la pregunta tres de la encuesta, donde los especialistas responden en qué medida consideran que los rasgos definidos son características reales de los proyectos.



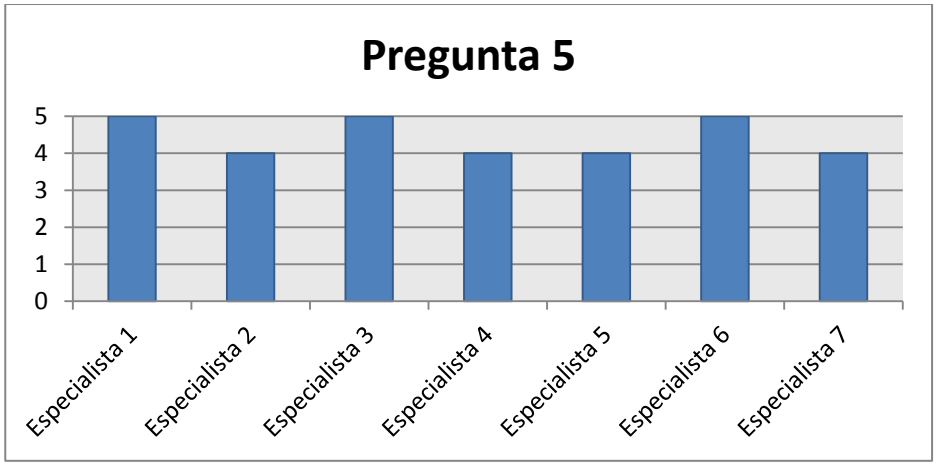
Gráfica 3 Respuesta a la pregunta 3 de la encuesta.

En la Gráfica 4 se encuentra la respuesta correspondiente a la pregunta cuatro, aquí los especialistas responden según sus conocimientos en qué medida consideran que si se aplica un Sistema Experto que utilice la base de casos propuesta se obtendrá una mejor clasificación de los proyectos.



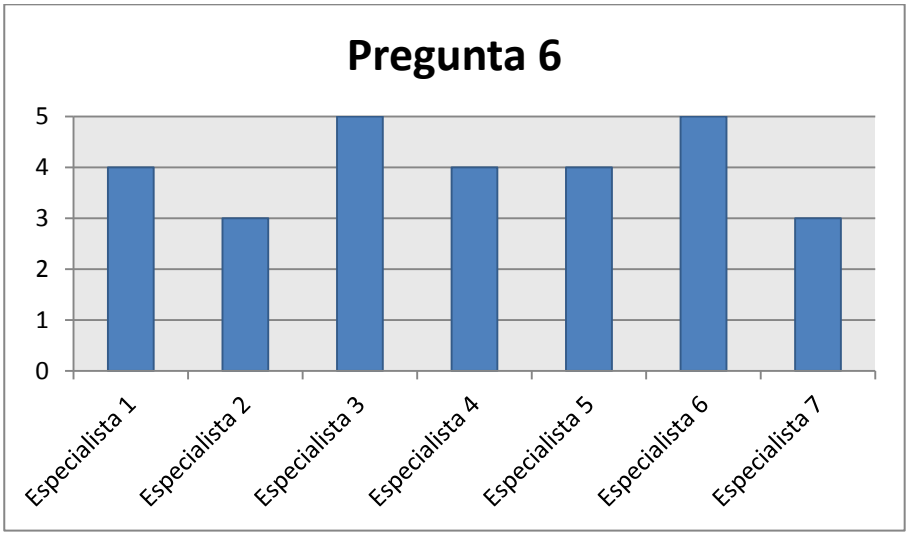
Gráfica 4 Respuesta a la pregunta 4 de la encuesta.

En la Gráfica 5 se encuentra la respuesta correspondiente a la pregunta cinco, donde el especialista debe de responder en qué medida considera que la estructura de los casos es la adecuada.



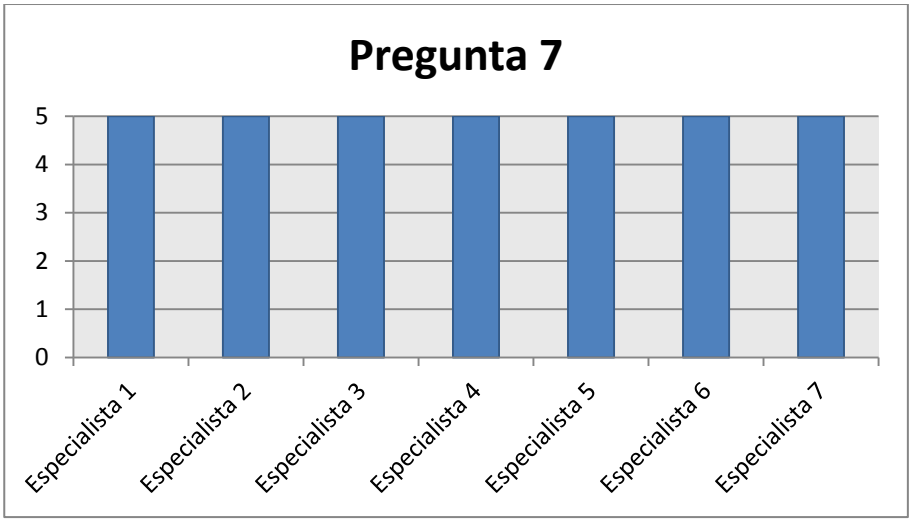
Gráfica 5 Respuesta a la pregunta 5 de la encuesta.

En la Gráfica 6 se encuentra la respuesta correspondiente a la pregunta seis, donde el especialista debe de responder en qué medida considera que la estructura de la base de casos es la adecuada.



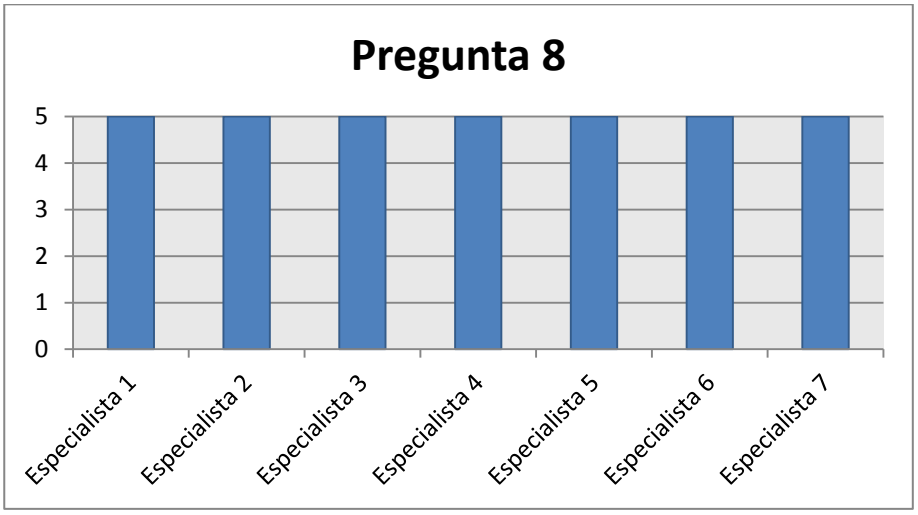
Gráfica 6 Respuesta a la pregunta 6 de la encuesta.

En la Gráfica 7 se encuentra la respuesta correspondiente a la pregunta siete, donde el especialista debe responder si considera útil el tratamiento de la incertidumbre.



Gráfica 7 Respuesta a la pregunta 7 de la encuesta.

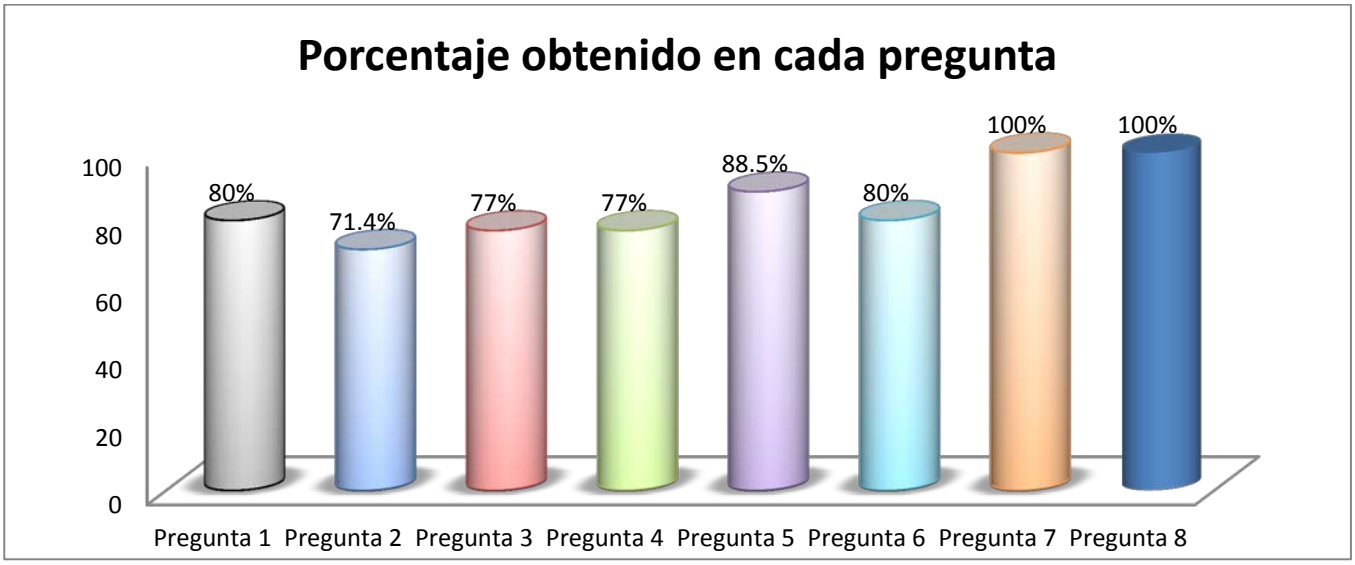
En la Gráfica 8 se encuentra la respuesta correspondiente a la pregunta ocho, donde el especialista responde de forma cualitativa en qué medida considera útil la utilización del umbral de semejanza.



Gráfica 8 Respuesta a la pregunta 8 de la encuesta.

2.5.4 Resultados Finales

El porcentaje de respuestas de los especialistas a cada una de las preguntas propuestas fue positivo, los resultados generales se pueden observar en la Gráfica 9.



Gráfica 9 Resultado de la encuesta.

Como se pudo observar en el gráfico, la pregunta dos relacionada con la clasificación propuesta es la que menor porcentaje tiene, debido a que no se tienen en cuenta la totalidad de los productos de software. Por

otra parte, se evidencia la total aprobación del tratamiento de la incertidumbre y del uso del umbral de semejanza. De manera general la aceptación de la propuesta de solución está por encima de la media, y el porcentaje faltante no es significativo. Por otra parte, se obtuvieron algunas sugerencias de los especialistas como: el uso de un número mayor de clasificaciones, así como de rasgos predictores, para que esté más completa la BC.

Conclusiones parciales

Se realizó el proceso de adquisición del conocimiento que arrojó como resultado la clasificación de los diferentes tipos de proyectos de desarrollo de software de acuerdo con el tipo de producto que genera; y se identificaron los rasgos y sus posibles valores. Se definió estructurar los casos a través de un conjunto de rasgos de la forma par atributo-valor y para la BC se usará la estructura plana. Se propuso el uso de las distancias de Hamming y Jaccard como funciones de semejanzas entre rasgos y se definió la función de semejanza entre casos teniendo en cuenta el grado de incertidumbre. Se definió el valor umbral de semejanza que debe tener un caso para ser considerado como parte de la solución de un problema. Finalmente, se realizó la validación de la propuesta de solución haciendo uso del método Delphi.

CAPÍTULO III: CARACTERÍSTICAS DEL SISTEMA

Introducción

En el presente Capítulo se describe la propuesta de solución al problema. Se detalla el modelo de dominio y se enumeran los requisitos funcionales y no funcionales, así como la descripción de los casos de uso del sistema y los diagramas relacionados.

3.1 Información que se maneja

La información que se maneja para el desarrollo de este trabajo está relacionada con las clasificaciones de los proyectos informáticos existentes, con respecto al producto final que se obtiene al finalizar el proyecto. Además, se utilizarán las características propias de cada producto.

3.2 Descripción de la solución propuesta

Con el fin de cumplir con los objetivos y requisitos planteados en este trabajo se tiene como propuesta de solución, el análisis y diseño de la base de casos para la posterior construcción de un sistema inteligente para la clasificación de proyectos.

3.3 Modelo de dominio

Una de las primeras actividades centrales de un ciclo de desarrollo consiste en crear un modelo conceptual para los casos de uso, en el cual se explican a sus creadores los conceptos significativos en un dominio del problema (25).

El Modelo de Dominio también conocido como Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Este representa clases conceptuales del dominio del problema, conceptos del mundo real y no de los componentes de software. Una clase conceptual también conocida como entidad puede ser una idea o un objeto físico (símbolo, definición y extensión). A continuación se presenta el modelo de dominio para el Sistema Inteligente de Clasificación de Proyectos.

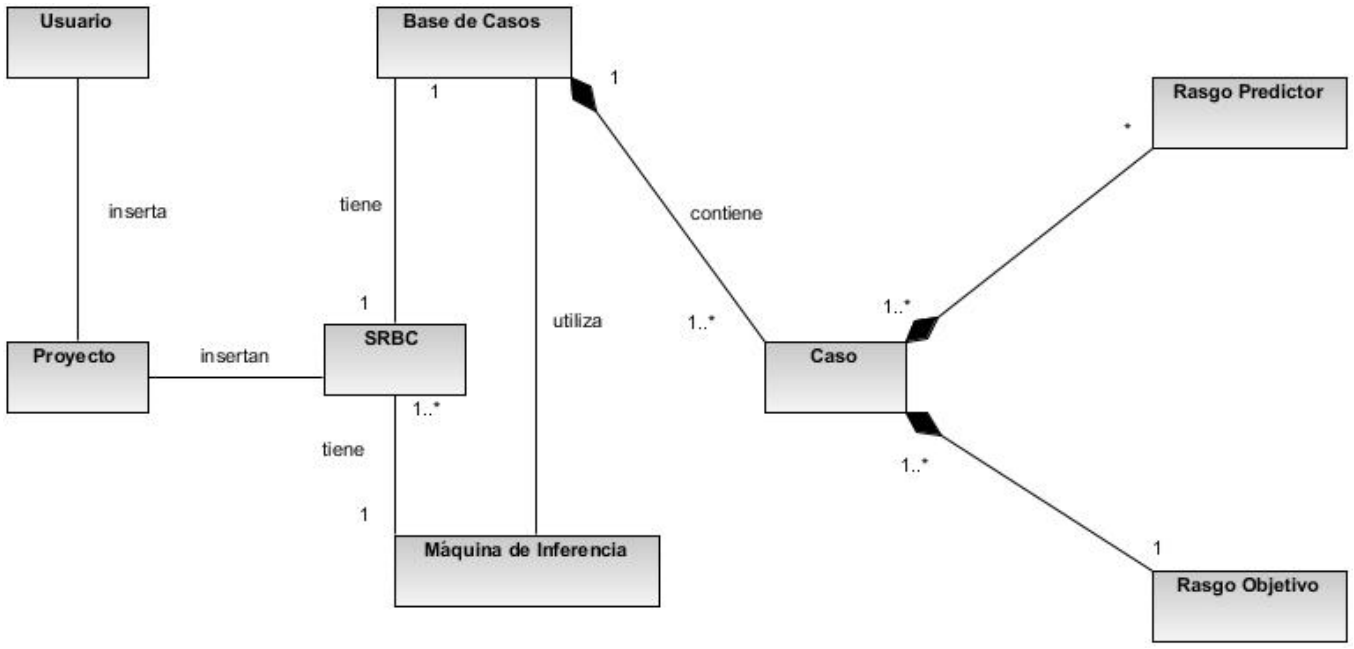


Figura 3 Modelo de Dominio. Sistema de Clasificación de Proyectos.

El usuario es el encargado de insertar la descripción de los proyectos que se van a clasificar. Una vez insertados, el Sistema de Razonamiento Basado en Casos o SRBC se encargará de manejar y visualizar la información al usuario, así como la respuesta y el almacenamiento de casos. El SRBC estará compuesto por la base de casos la cual posee un conjunto de casos y sus correspondientes rasgos predictores y objetivos; y la máquina de inferencia, en la cual se implementan los mecanismos de inferencia haciendo uso de los casos de la base de casos.

3.4 Especificación de las funcionalidades

Los requisitos deben especificarse antes de intentar comenzar la construcción del producto, sin ellos no podrá ser posible llevar a cabo las etapas de diseño y construcción correctamente.

3.4.1 Requisitos Funcionales

Los Requisitos Funcionales (RF) de un sistema describen lo que el sistema debe hacer. Estos dependen del tipo de software que se desarrolle y de los posibles usuarios del sistema (40). A continuación se detallan los RF que el sistema debe cumplir:

RF 1: Autenticar Usuario.

- Usuario.
- Contraseña.

RF 2: Insertar un Usuario.

- Nombre.
- Nombre del Usuario.
- Contraseña.
- Rol.

RF 3: Modificar Usuario (Remitirse al RF2).

RF 4: Eliminar Usuario.

- Nombre de Usuario.

RF 5: Mostrar Usuario.

RF 6: Insertar Proyecto.

- Nombre del Proyecto.
- Facultad.
- Centro.
- Objeto social.
- Tipo de datos.
- Diseño arquitectónico.
- Patrones.
- Lenguaje de programación.
- Consumo de recursos.
- Tecnología utilizada.
- Características especiales de seguridad.

RF 7: Modificar Proyecto (Remitirse al RF6).

RF 8: Eliminar Proyecto.

- Nombre.
- Centro.

RF 9: Mostrar Proyectos.

RF 10: Realizar Clasificación.

- Proyecto (Remitirse al RF6).
- Rasgo (Remitirse al RF12).
- Clasificaciones (Remitirse al RF15).

RF 11: Mostrar Solución.

RF 12: Insertar Rasgo.

- Nombre.
- Objeto social.
- Tipo de datos.
- Diseño arquitectónico.
- Patrones.
- Lenguaje de programación.
- Consumo de recursos.
- Tecnología utilizada.
- Características especiales de seguridad.

RF 13: Eliminar Rasgo.

- Nombre.

RF 14: Mostrar Rasgo.

RF 15: Modificar Rasgo (Remitirse al RF12).

RF 16: Insertar Clasificaciones.

- Nombre.
- Rasgo.
- Descripción.

RF 17: Modificar Clasificaciones (Remitirse al RF16).

RF 18: Eliminar Clasificaciones.

- Nombre.
- Rasgo.

RF19: Mostrar Clasificaciones.

RF20: Insertar Caso.

- Nombre.

- Rasgos.

RF21: Elimina Caso.

- Nombre.

RF22: Modificar Caso (Remitirse al RF20).

RF23: Mostrar Caso.

3.4.2 Requisitos no Funcionales

Los Requisitos No Funcionales (RNF) como su nombre sugiere, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes que este debe tener (40). Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Constituyen elementos importantes para que el producto final satisfaga las expectativas de los clientes. A continuación se enumeran los RNF:

Requisitos de Apariencia o interfaz externa:

- El sistema brindará una interfaz amigable para sus usuarios, con predominio de color gris en diversas tonalidades y azul. Las letras serán legibles con formato Arial de 12 píxeles, de color azul, blanco y negro.

- Los errores que sean visibles al usuario, se marcarán con un asterisco de color rojo y deben señalar las posibles causas y sus soluciones.

Requisitos Software.

- La computadora donde se ejecute la aplicación debe tener instalado Firefox 3.X o superior, Chrome 5.X o superior o Safari, debido a que los estándares de CSS3 no son reconocidos por muchas de las versiones de Internet Explorer (6.X, 7.X). Debe tener alojado el SGBD PostgreSQL 8.4 o contar con dicho servidor externo. En caso de acceder desde Internet Explorer se mostrará un mensaje de información indicando que debe utilizar un explorador diferente como los propuestos.

- En caso de ser externo el servidor de base de datos, debe tener instalado PostgreSQL 8.4.

- El servidor de aplicaciones debe tener instalado el Apache Tomcat como servidor web y la JVM.

Diseño.

- El sistema implementado será una aplicación web.
- El sistema se implementará usando la plataforma JAVA.
- El sistema estará basado en un estilo arquitectónico en capas. Las capas estarán distribuidas de la siguiente manera:
    - Capa web: En esta capa se encuentra la lógica de presentación se maneja todo el flujo web utilizando la implementación del patrón Modelo-Vista-Controlador que brinda el framework Grails. Y por otra parte las vistas, que son los recursos que junto al modelo generado por los controladores le permiten al cliente visualizar la información.

    - Capa de dominio: Contiene las entidades que persistirán en la base de datos, además encapsula toda la lógica relacionada con el dominio que abarca el negocio. Esta brinda las funcionalidades mediante fachadas de servicio que son utilizadas por los controladores en la capa web y se implementan procesos de negocio que perduran en el tiempo.

    - Capa de datos: Encapsula la lógica de acceso a datos abstrayendo a las capas superiores del mecanismo de acceso a la base de datos. Dicha abstracción se realiza a través del framework de persistencia GORM, que implementa Grails.

Requisitos de Seguridad.

- Seguridad y control a nivel de usuarios y contraseñas, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realicen. Las contraseñas sólo podrán ser cambiadas por el propio usuario o por el administrador informático del sistema.

Requisitos de Fiabilidad.

- El sistema debe de estar disponible las 24 horas del día, incluyendo días feriados y fines de semana.

- El sistema tendrá un respaldo de la información de la base de casos, permitiendo la recuperación ante la pérdida parcial o total de la información.

Requisitos de Eficiencia.

- El sistema procesará una transacción en un tiempo no mayor a 1 segundo, a partir de que la petición se recibe en el servidor, es decir, esta cifra no incluye los retardos por concepto de tráfico de red.

3.5 Diagrama de Casos de uso del Sistema

RUP define un caso de uso como un conjunto de instancias, donde cada instancia es una secuencia de acciones que lleva a cabo un sistema que produce un resultado observable de valor para un actor concreto. Y el diagrama de casos de uso es un modelo de las funciones deseadas para el sistema y su entorno, este sirve como contrato entre el cliente y los desarrolladores. Se utiliza como entrada esencial para las actividades de análisis, diseño y prueba (41).

3.5.1 Definición de los actores del sistema

Actores	Justificación
Usuario	Este actor debe autenticarse para acceder al sistema. Inserta la descripción de los proyectos y obtiene su clasificación.
Experto	Es una especialización del actor Usuario, es el encargado de insertar, actualizar, eliminar los rasgos, los casos y las clasificaciones.
Administrador	Es el encargado de insertar, modificar y eliminar los usuarios del sistema, además de realizar todas las acciones del usuario y el experto.

Tabla 6 Actores definidos.

3.5.2 Diagrama de casos de uso del sistema

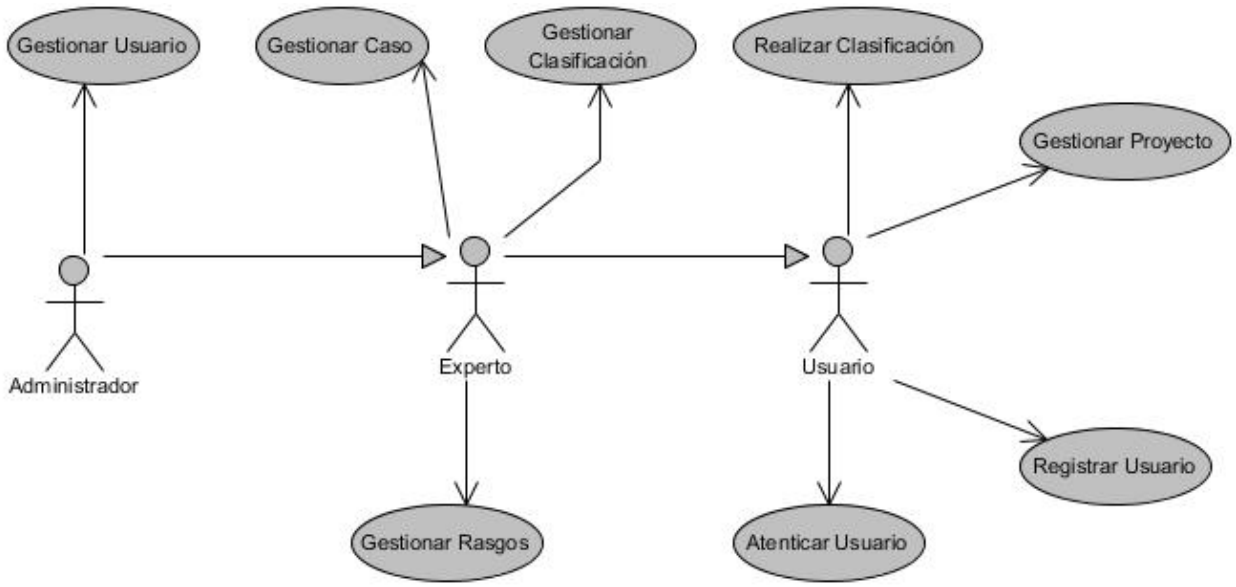


Figura 4 Diagrama de Casos de Uso del Sistema.

Como se muestra en la figura anterior el Administrador tendrá la capacidad de realizar todas las acciones que brinda el sistema, y los demás usuarios podrán realizar sólo las acciones que sus niveles de seguridad le confieran.

3.5.3 Descripción de los Casos de uso del Sistema

Un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema, estos agrupan los requisitos funcionales con los que debe cumplir el sistema. La descripción de cada caso de uso permitirá obtener una visión más cercana del sistema, consolidándose este objetivo con los prototipos de interfaz de usuario. A continuación se muestra la descripción del CU Gestionar Caso. Para consultar las descripciones correspondientes a otros casos de uso ver Anexo 6.

Caso de Uso Gestionar Caso

Caso de Uso	**Gestionar Caso.**
Actor	Experto, Administrador.
Resumen	El sistema debe permitir insertar, modificar y eliminar un caso.
Precondiciones	El usuario debe estar registrado.

Prioridad	Crítico.
Flujo Normal de Eventos	
Sección "Insertar Caso"	
Acciones del Actor	**Respuestas del sistema**
1-El Administrador solicita la opción de Insertar Caso.	2-El sistema muestra la interfaz Insertar Caso. Se solicitan los datos. (Remitirse al RF 12).
3-El Administrador introduce los datos solicitados por el sistema.	4-El sistema verifica que los datos sean correctos. 5-El sistema guarda los datos del nuevo caso. 6-El sistema informa que el caso se creó correctamente.
Prototipo de Interfaz	

Flujo alterno	
4-Existen datos incorrectos	
Acciones del Actor	**Respuestas del sistema**
	4.1 El sistema muestra un mensaje de error informando al usuario que los datos son incorrectos. Regresa a la acción 2.
Prototipo de Interfaz	

Pos-condiciones: Se crea un nuevo caso.

Sección "Modificar Caso "

Acciones del Actor	Respuestas del sistema
1-El Administrador solicita la opción Modificar Casos.	2-El sistema muestra la interfaz Modificar Casos. Se solicitan los datos de los rasgos a modificar.
3-El Administrador introduce los datos solicitados por el sistema. 6-El Administrador inserta los datos solicitados.	4. El sistema busca y comprueba el caso a modificar. 5. El sistema muestra los datos del caso a modificar. 7- El sistema comprueba que los datos sean correctos. 8-El sistema modifica el caso. 9- Si los datos son correctos el sistema actualiza los datos del caso y notifica mediante un mensaje que el caso fue actualizado correctamente.

Prototipo de Interfaz

Flujo alterno

4-No existe el caso.

Acciones del Actor	Respuestas del sistema
	4.1 El sistema muestra un mensaje de error informando que no existe el caso que se desea modificar. Regresa a la acción 2.

Prototipo de Interfaz

7-Existen datos incorrectos.

Acciones del Actor	Respuestas del sistema

	7.1 El sistema muestra un mensaje de error informando que los datos son incorrectos. Regresa a la acción 5.
Prototipo de Interfaz	
	
Pos-condiciones	Se actualiza el caso.
Sección “Eliminar Caso”	
Acciones del Actor	**Respuestas del sistema**
1-El Administrador solicita la opción Eliminar Caso.	2-El sistema muestra la Interfaz para Eliminar un caso. Se solicitan los datos del caso a eliminar.
3-El Administrador introduce los datos solicitados por el sistema.	4-El sistema busca el caso a eliminar. 5-El sistema muestra un mensaje de confirmación.
Prototipo de Interfaz	

Flujo alterno	
5-Confirmar eliminación del caso.	
Acciones del Actor	**Respuestas del sistema**
5. a.1-Selecciona la opción de eliminar.	5. a.2 El sistema muestra un mensaje confirmando que el caso se eliminó correctamente.

Prototipo de Interfaz

5-Cancelar eliminación del caso.	
Acciones del Actor	**Respuestas del sistema**
5.a.3-Selecciona la opción de No.	Regresa a la acción 2.

Pos-condiciones	Se elimina el caso.

Conclusiones parciales

En este capítulo se elaboró el Modelo de Dominio, en el cual se tratan los conceptos fundamentales del sistema, así como la relación que existe entre los mismos. Se definieron los requisitos funcionales y no funcionales que debe cumplir el sistema. Se confeccionó el diagrama de Casos de Uso del Sistema teniendo en cuenta las funcionalidades que debe realizar cada actor identificado y se realizó la descripción textual de cada caso de uso definido.

CAPÍTULO IV: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En este capítulo se expone el análisis y diseño para dar solución al problema planteado, modelándose los artefactos necesarios que contribuirán a la posterior implementación del sistema. Entre los artefactos a desarrollar se encuentran los diagramas de clases del análisis y del diseño. Se analiza el patrón arquitectónico a utilizar.

4.1 Análisis

En el análisis se examinan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El lenguaje que se utiliza en el análisis se basa en un modelo de objetos conceptual, llamado modelo de análisis. El modelo de análisis es la primera representación técnica de un sistema y debe lograr dos objetivos primarios: describir lo que requiere el cliente y establecer una base para la creación de un diseño de software (42). Esencialmente el objetivo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos, que sea fácil de mantener y que ayude a estructurar el sistema entero. Proporciona una estructura centrada en el mantenimiento, y en aspectos tales como la flexibilidad ante los cambios y la reutilización.

En el análisis se definen todas las clases que son relevantes al problema que se va a resolver, las operaciones, las relaciones y comportamientos asociados con ellas. Este modelo es usado para representar la estructura global del sistema, describe la realización de casos de uso. El modelo de análisis se describe utilizando el lenguaje de los desarrolladores, y puede por tanto introducir un mayor formalismo y ser utilizado para razonar sobre los funcionamientos internos del sistema. Esboza como llevar a cabo las funcionalidades del sistema y sirve como una primera aproximación al diseño.

Diagramas de clases del análisis

Uno de los principales artefactos del análisis es el diagrama de clases de análisis, en él se representan los conceptos en un dominio del problema, además se representan las clases de análisis y sus relaciones entre sí. Las clases de análisis representan abstracciones de una o varias clases y/o subsistemas del

diseño del sistema, las cuales se caracterizan por centrarse en el tratamiento de los requisitos funcionales. Las clases del análisis se clasifican en:

- **Clase Interfaz**: Modela la interacción entre el sistema y sus actores. A menudo, representan abstracciones de ventanas, formularios, entre otros. Cada clase interfaz debe estar relacionada con al menos un actor.
- **Clase Entidad**: se utiliza para modelar información que posee una larga vida y que es a menudo persistente.
- **Clase Control**: Representa coordinación, secuencia, transacciones, control de otros objetos y a menudo encapsula a un caso de uso en concreto.

A continuación se muestra el diagrama de clases del análisis del CU Gestionar Clasificación. Para consultar los diagramas de clases correspondientes a otros casos de uso ver Anexo 7.

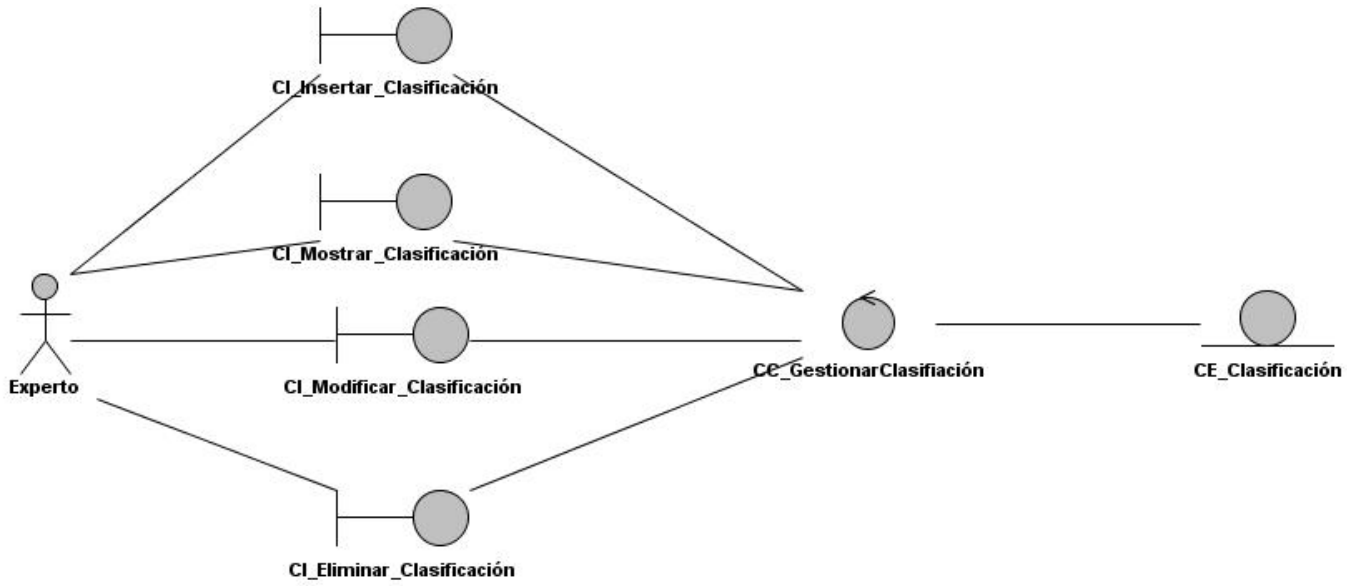


Figura 5 Diagrama de análisis del CU Gestionar Clasificación.

En el diagrama anterior se identifican 4 clases interfaz (CI_Insertar_Clasificación, CI_Mostrar_Clasificación, CI_Modificar_Clasificación y CI_Eliminar_Clasificación) que permitirán la

interacción entre el sistema y su actor. La clase controladora (CC_GestionarClasificación) que será la encargada de controlar las funcionalidades y la clase entidad (CE_Clasificación) que se utiliza para representar las clasificaciones que pueden ser dadas a los proyectos.

4.2 Diseño

El diseño es el centro de atención final de la fase de elaboración y el comienzo de las iteraciones de construcción. Este contribuye al desarrollo de una arquitectura estable y sólida y a crear un plano del modelo de implementación. En el diseño se modela el sistema y se encuentra una forma para que soporte todos los requisitos (42). Los propósitos del diseño son crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando requisitos o subsistemas individuales, interfaces y clases. Debe ser capaz de descomponer el trabajo de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.

4.2.1 Patrones

Los patrones son soluciones listas para usar, aplicables a problemas que se repiten con frecuencia en contextos acotados. Ayudan a construir la experiencia colectiva de Ingeniería de Software, son una abstracción de "problema – solución", ocupándose de problemas recurrentes, identificando y especificando abstracciones de niveles más altos que componentes o clases individuales y proporcionando vocabulario y entendimiento común (43). Existen diferentes tipos de patrones, entre los que se encuentran los patrones de arquitectura y los patrones de diseño.

Patrón de arquitectura

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. Es un nivel de abstracción mayor que el de los Patrones de Diseño, que solo se relaciona con los aspectos del diseño.

En el Capítulo I se propuso el uso del framework Grails. Como se mencionó anteriormente las aplicaciones desarrolladas con este framework utilizan el patrón arquitectónico MVC (Modelo-Vista-Controlador). La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. En este caso el modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. La vista es la encargada de

recibir los datos del modelo y lo muestra al usuario; y el controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

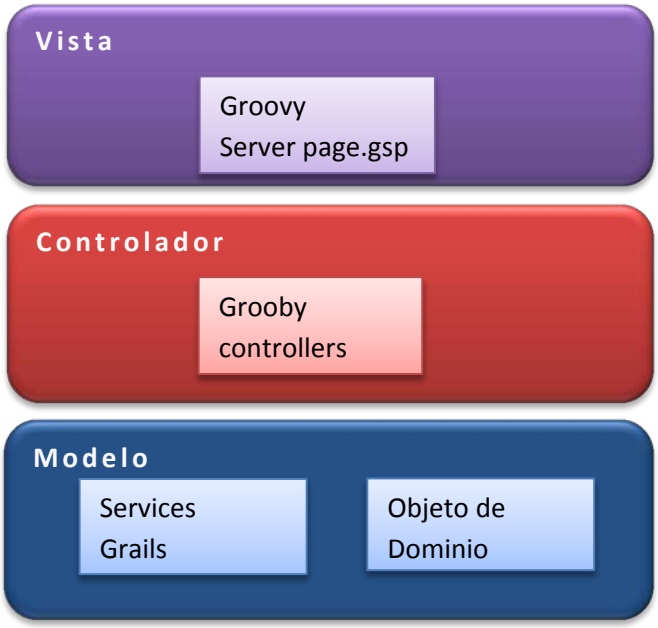


Figura 6 Modelo-Vista-Controlador.

Este patrón presenta varias ventajas, como por ejemplo el soporte de múltiples vistas. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes. Otra ventaja es la adaptación al cambio, ya que los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Diagramas de clases del diseño

A través del flujo de diseño, uno de los artefactos más importantes a obtener son los diagramas de clases del diseño, donde se exponen las clases que intervienen en las realizaciones de los casos de uso del sistema. En este tipo de diagrama se presenta un nivel de detalle más alto que los diagramas de clases

del análisis Los diagramas de clases de diseño describen gráficamente las especificaciones de las clases del software y contienen las clases, atributos, métodos, navegabilidad y dependencia existentes entre ellas. A continuación se muestra el diagrama de clases del diseño del CU Realizar Clasificación. Para consultar los diagramas de clases del diseño correspondientes a otros casos de uso ver Anexo 8.

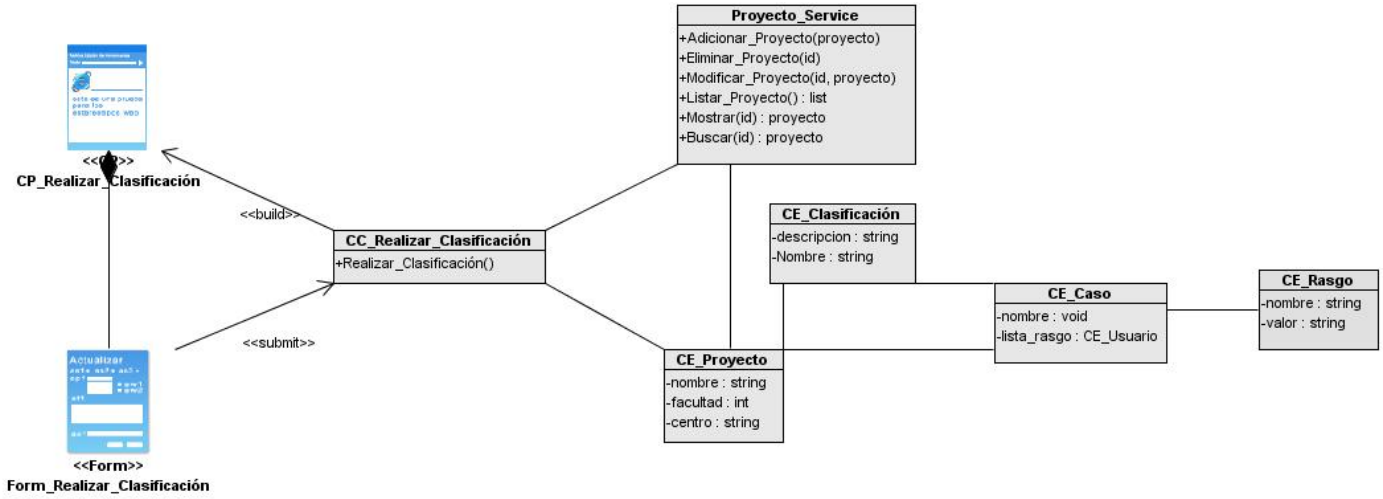


Figura 7 Diagrama del diseño del CU Realizar Clasificación.

La imagen anterior muestra el diagrama de clases del diseño del caso de uso Realizar Clasificación, que evidencia las especificaciones de las clases del software y de las interfaces del mismo; representando sus métodos, atributos y dependencias. Como se puede apreciar se contará con una interfaz (CP_Realizar_Clasificación) la cual será el vínculo entre el usuario y el sistema, la clase controladora que será la encargada de gestionar la información y de interactuar con las entidades, y en la clase Service se implementan las funcionalidades.

Conclusiones parciales

El análisis y diseño de un software brinda la primera visión de lo que pudiera ser la solución en el desarrollo del mismo. En este capítulo se han mostrado las funcionalidades que tendrá el sistema, a través de una serie de modelos que permiten tener un mayor entendimiento del mismo. Se definieron las principales clases del análisis y el diseño mostrando sus relaciones mediante diferentes diagramas. El resultado del capítulo posibilitará la guía para la futura implementación de la aplicación.

CONCLUSIONES GENERALES

Se diseñó un sistema inteligente capaz de apoyar el proceso de clasificación de los proyectos de desarrollo de software, teniendo en cuenta el producto final que se genera en los mismos. Esto proporcionará valores más acertados de estimación del tiempo de duración de los proyectos, ya que se trabajará sobre la base de proyectos similares. La propuesta de solución fue desarrollada haciendo uso de una de las técnicas más utilizadas actualmente para apoyar el proceso de toma de decisiones: el razonamiento basado en casos. Con esta técnica los problemas se resuelven de manera similar a como lo harían los expertos humanos en el dominio. Se propusieron las estructuras que conformarán la base de casos y un conjunto de algoritmos para recuperar los casos, con el objetivo de determinar las posibles soluciones o caracterizar los proyectos. Se representan los diagramas de clases del análisis y de clases del diseño, siendo estos el resultado de la transformación de los requisitos del software. Se logró la creación de las bases para la futura implementación del sistema.

RECOMENDACIONES

Con el objetivo de obtener una solución más completa y darle continuidad al mismo, se presentan un conjunto de recomendaciones que se deberían tener en cuenta la posterior implementación del trabajo:

- Investigar y tener en cuenta otras clasificaciones de los proyectos informáticos teniendo en cuenta el criterio escogido.
- Investigar y lograr el aumento del número de rasgos predictores que permitan realizar correctamente la clasificación.
- Desarrollar el módulo de adaptación, revisión y aprendizaje.

REFERENCIAS BIBLIOGRÁFICAS

1. Eduteka. *Informática educativa.* [En línea] [Citado el: 12 de 11 de 2011.] http://www.eduteka.org/proyectos.php/1/2500.

2. **Marcelo Claudio Périssé.** Ciencia y Técnica Administrativa. [En línea] 2008. [Citado el: 08 de 09 de 2011.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c1/c1.htm.

3. **Lío, Daniel Galvez.** *Curso de sistemas basados en el conocimiento.* Universidad de las Villas, Cuba : s.n., 1998.

4. *Sistemas de razonamiento basado en casos aplicado a sistemas de líneas de productos software.* **Augusto Cortez Vásquez, Carlos Navarro Depaz, Jaime Pariona Quispe.** s.l. : Revista de Investigación de Sistemas e Informática. , 2010. 1815-0268.

5. *SISTEMAS BASADOS EN EL CONOCIMIENTO.* **GARCÍA, MARILUZ ROMERO y RODRÍGUEZ, JORGE ENRIQUE RODRÍGUEZ.** Colombia : Fondo De Publicaciones De La Universidad Distrital, 2004, Vol. I. 1794-211X.

6. **Colombia, Universidad de.** Inteligencia Artificial. *Sistemas Basados en Conocimiento.* [En línea] [Citado el: 26 de 10 de 2011.] http://disi.unal.edu.co/~lctorress/iartificial/IA0011l.pdf.

7. **Producción., Departamento General de.** *Áreas temáticas y de investigación.*

8. **Kenia Labori de la Rosa, Izary Rondón Robaúl.** *Metodologías para el desarrollo de software multimedia:Análisis comparativo y propuesta.* Habana : s.n., 2007.

9. Qué es, Significado y Concepto. *Definicion.de.* [En línea] 2008-2012. [Citado el: 2 de Mayo de 2012.] http://definicion.de/proyecto/.

10. **Vismar G. Flores, Roberto C. Cueva.** *INTRODUCCIÓN A LOS CONCEPTOS DE GESTIÓN DE PROYECTOS DE SOFTWARE.*

11. Proyectos Informáticos. [En línea] http://www.slideshare.net/mcedenho/proyectos-informaticos2.

12. Proyectos de uso público y proyectos de uso privado. [En línea] http://es.scribd.com/doc/57171848/PROYECTO-DE-USO-PUBLICO-Y-PROYECTO-DE-USO-PRIVADO.

13. **García, Carlos Martín.** *Estudio de la Clasificación de los proyectos informáticos y del Establecimiento de su estructura de descomposición del trabajo.* 2004.

14. **Munárriz, Luis Álvarez.** *Fundamentos de Inteligencia Artificial.* Murcia : Secretariado de Publicaciones, 1994. 84-7684-563-4.

15. Universidad Nacional de Colombia. *Sistemas Basados en el Conocimiento.* [En línea] [Citado el: 19 de Abril de 2012.] http://dis.unal.edu.co/profesores/lucas/iartificial/IA0011l.pdf.

16. *El Razonamiento Basado en Casos en el ámbito de la Enseñanza/Aprendizaje.* **Martínez Sánchez, MSc. Natalia, Ferreira Lorenzo, Dra. Gheisa y García Lorenzo, Dra. María M.** 25-32, s.l. : Revista de Informática Educativa y Medios Audiovisuales, 2008, Vol. V. 1667-8338.

17. *Case-Based Reasoning II. A continuation of our Overview of the Hottest New Approach to Knowledge-Based Systems Development.* **Harmon, P.** 12, s.l. : Intelligent Software Strategies, 1991, Vol. 7.

18. *Un sistema de razonamiento basado en casos para la clasificación de fallos en sistemas dinámicos.* **Bregón, Anibal, y otros, y otros.** Valladolid : Actas del III Taller Nacional de Minería de Datos y Aprendizaje, 2005. 84-9732-449-8.

19. **Gómez, Fernando Díaz.** *Razonamiento Basado en Casos (CBR). Introducción.* Universidad de Valladolid : E. U. de Informática – Segovia.

20. **Mora, A.** *Modelos de Organización de Memoria.* 2008.

21. Diccionario. *Significado de incertidumbre.* [En línea] http://es.thefreedictionary.com/incertidumbre.

22. **Gutiérrez Martínez, Iliana, Bello Pérez, Rafael E. y Tellería Rodríguez, Andrés.** *UN SISTEMA BASADO EN CASOS PARA LA TOMA DE DECISIONES EN CONDICIONES DE INCERTIDUMBRE.* Santa Clara, Cuba : s.n., 2002.

23. **Lozano, Laura y Fernández, Javier.** *Razonamiento Basado en Casos: Una Visión General.* 2005.

24. **VÁSQUEZ, MÓNICA JOSÉ BAUTISTA.** *PROTOTIPO DE SISTEMA EXPERTO LEGISLATIVO:VERIFICACIÓN DE CONSTITUCIONALIDAD O INCONSTITUCIONALIDAD EN INICIATIVAS DE LEY, BASADO EN LA CONSTITUCIÓN DE LA REPÚBLICA DE GUATEMALA.* Guatemala : s.n., 2005.

25. **Jacobson, Ivar, Rumbaugh, James y Booch, Grady.** *El lenguaje Unificado de Modelado. Manual de Referencia.* Madrid : Addison Wesley, 2007.

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

27. **Jacobson, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid-España : Imprenta FARESO, S.A, 2000.

28. **larman, Craig.** *UML y Patrones: Una introducción al análisis y el diseño orientado a objetos y al proceso unificado.* 2009.

29. **Abdul-Jawad, B.** *Groovy and Grails Recipes.* New York, NY. : Apress, Inc., 2009.

30. **Christopher Judd, James Shingler.** *Beginning Groovy and Grails.* New York, NY. : Apress, Inc, 2008.

31. **Dickinson, J.** *Grails 1.1 Web Application Development.* Birmingham : Packt Publishing, 2009.

32. **Postgresql., El equipo de desarrollo de.** Postgresql. [En línea] http://www.postgresql.org.

33. **Pressman, Roger.** *Ingeniería del Software. Un enfoque práctico.* 2002.

34. **Morales, Irene Suárez y Castro, Elizabeth Rodríguez.** *Método para evaluar la calidad de la arquitectura en cuanto a Requisitos No Funcionales en los software de Realidad Virtual.* Habana : s.n., 2009.

35. *Memoria Organizacional Basada en Casos.* **Perez, Alonso.** Recife, Brasil. : Revista de Ciencia, 2002.

36. **Martin, Javier y López, Pliego.** *Introducción a la Estadísticas y Economía Empresarial 3 Edición.* 2005. ISBN: 84-9732-316-5.

37. **RODRÍGUEZ, Ing. SERGIO MICHEL RIVERA.** *MODELO DE UN SISTEMA DE RAZONAMIENTO BASADO EN CASOS PARA EL ANÁLISIS EN LA GESTIÓN DE RIESGOS.* Habana : s.n., 2010.

38. *Modelo para diseñar sistemas de enseñanza-aprendizaje inteligentes utilizando el razonamiento basado en casos.* **Martínez Sánchez, Dra. Natalia, García Lorenzo, Dra. María Matilde y García Valdivia, Dra Zoila Zenaida.** No. 3, s.l. : Avances en Sistamas e Informática, 2009, Vol. Vol 6. ISSN 1657-7663.

39. **Astigarraga, Eneko.** Método Delphi. [En línea] 2007. [Citado el: 2012 de mayo de 24.] http://www.unalmed.edu.co/~poboyca/documentos/documentos1/documentos-Juan%20Diego/Plnaifi_Cuencas_Pregrado/Sept_29/Metodo_delphi.pdf.

40. **Sommerville, Ian.** *Ingeniería de Software.* Madrid : Pearson Educación, SA., 2005. 84-7829-074-5.

41. **IBM.** *Rational Unified Process for SOMA.* 2007.

42. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, 2000. 84-7829-036-2.

43. **Villa, Madelys Cuesta.** *Modelo para la ayuda a la toma de decisiones en la selección de patrones de desarrollo de software.* Habana : s.n., 2007.