# Universidad de las Ciencias Informáticas

## Facultad 2


# Título: Base de Casos para estimar la duración de un proyecto de desarrollo de software

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

**Autor(es):** Pedro Carlos Abreu Jiménez

Jeny Del Sol Chang

**Tutor(es):** MsC. Yadira Ruiz Constanten

**Co-tutor:** Ing. Dasiel Cordero Morales.

Ciudad de La Habana, Junio del 2012

"Año 53 de la Revolución"

*"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica."*

*Aristóteles*

# AGRADECIMIENTOS

DECLARACION DE AUTORIA

Declaramos ser los únicos autores de este trabajo y autorizamos al Centro ISEC de la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio.

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

| | |
|---|---|
| _____ | _____ |
| Pedro Carlos Abreu Jiménez | Jeny del Sol Chang |
| _____ | _____ |
| MsC. Yadira Ruiz Constanten | Ing. Dasiel Cordero Morales |

**RESUMEN**

La Universidad de las Ciencias Informáticas, a casi 10 años de su fundación, ha logrado adquirir experiencia paulatinamente en la realización de proyectos de desarrollo de software. Sin embargo, no se almacena la información relacionada con los procesos de gestión del tiempo y todos los aspectos que influyen sobre la duración de este, incluyendo el uso de las nuevas tecnologías, para ser utilizadas luego en próximas estimaciones.

El presente trabajo de diploma pretende mediante el diseño de una base de casos, representar la información que pueda ser utilizada para estimar la duración de proyectos de desarrollo de software en la universidad. Aprovechando para ello la experiencia y los conocimientos que sobre la estimación de tiempo de proyectos de desarrollo de software tienen los especialistas vinculados a esas tareas.

La investigación parte de la identificación de indicadores que permiten apoyar la determinación de tamaño y esfuerzo del desarrollo de un proyecto de software. Formalizándolos luego en un modelo computacional en el que se aplican técnicas planteadas por los sistemas basados en conocimiento y se implementa una solución que permite almacenar la experiencia y los conocimientos.

**PALABRAS CLAVES**

Base de Casos, Función de Semejanza, Indicadores, Métodos de Estimación, Umbral

# ÍNDICE DE TABLA

# ÍNDICE DE TABLA

# ÍNDICE DE TABLA

**ÍNDICE DE TABLAS.**



**ÍNDICE DE FIGURAS.**

## INTRODUCCIÓN

La Universidad de las Ciencias Informáticas (UCI) tiene como misión formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática. (1) El cumplimiento en tiempo de los compromisos pactados tiene un gran peso en esta industria.

Pese a que se han desarrollado diversos métodos para realizar la estimación no siempre se logra concluir un proyecto en el tiempo previsto. Como resultado de la planificación que se realiza al comenzar el proyecto se obtiene la estimación de este tiempo, pero en este momento no están definidos todos los elementos que se van a desarrollar, lo que dificulta el proceso.

En un proyecto de desarrollo de software, en los que se incluyen las tecnologías de software, es aun más difícil realizar una planificación. El objetivo de la planificación del proyecto es proporcionar un marco de trabajo que permita hacer estimaciones razonables de recursos, costo y planificación temporal. (2) La dificultad en la estimación de proyectos de desarrollo de software radica en que el producto final es bastante intangible y su valor real depende no sólo de su correcta realización, sino además del momento en que se pone en servicio, de la calidad apreciada por el usuario, y de su adaptabilidad, mantenimiento y extensibilidad. Estudios realizados entre el 2007 y el 2008 por especialistas del Centro para la Excelencia en el Desarrollo de Proyectos Tecnológicos (Calisoft) en la UCI muestran la incoherencia entre la estimación y los resultados finales de los proyectos que desarrolla esta institución. Esto se debió en gran medida a que en la fase de inicio se estimó el tiempo del levantamiento de requisitos y prototipo, sin embargo quedó demostrado que las actividades anteriores no eran suficientes y que debían ser completadas con la descripción del negocio, tarea que les permitió ganar en entendimiento y claridad. En estos proyectos no se tuvo en cuenta el impacto de los riesgos como por ejemplo el nivel de experiencia que tenían los desarrolladores, aspecto importante para el desarrollo del proyecto. Otro aspecto que influyó fue la inadecuada estimación de los procesos a automatizar causando un atraso de más del doble de lo estimado. (3)

El conjunto de capacidades organizativas, técnicas y financieras de liderazgo utilizadas para alcanzar los objetivos de un proyecto se define como gestión de proyectos. Esta trata de obtener un resultado

previamente definido en el contexto de un programa dado, de unos costes específicos y con la calidad requerida (4). En la actualidad se pueden encontrar diferentes escuelas u organizaciones dedicadas a guiar el proceso de administración y gestión de proyectos entre ellas se encuentran: PMI (Project Management Institute), PRINCE2 (Projects in Controlled Environments), P2M (Project & Program Management for Enterprise Innovation) y V-Model.

PMI es una asociación sin fines de lucro, de las más grandes del mundo en el área de administración de proyectos, reconocida a nivel mundial y cada vez más en Latinoamérica, además es líder en desarrollo de nuevos estándares para la profesión a nivel global. PMI fija los estándares profesionales por medio de su Guía para el Cuerpo de Conocimiento en Administración de Proyectos (PMBOK por sus siglas en inglés “A Guide to the Project Management Body of Knowledge”). (5)

El PMBOK es una colección de procesos y áreas de conocimiento generalmente aceptadas como las mejores prácticas dentro de la gestión de proyectos que provee los fundamentos que son aplicables a un amplio rango de proyectos, incluyendo los de construcción, software e ingeniería. PMBOK reconoce cinco grupos de procesos básicos y nueve áreas de conocimiento comunes a casi todos los proyectos. Estas áreas de conocimientos, tienen una serie de elementos de entrada a los que se le aplican técnicas y herramientas para obtener una salida. (6)

El área de Gestión del Tiempo del Proyecto descrita por PMBOK es quien obtiene como salida una estimación temporal de la duración del proyecto y es la guía de muchos de los proyectos del mundo principalmente en América. Una de las técnicas de esta área de conocimientos para estimar la duración de las actividades es el Juicio de Expertos.

Los juicios emitidos por los expertos son avalados por su experiencia en entornos similares apoyados en datos obtenidos de proyectos anteriores y almacenados. Las principales desventajas son el alto costo en tiempo y recursos humanos necesarios para su implantación, así como la subordinación al nivel de experiencia y conocimientos en el entorno que puedan aportar los expertos. (7)

La evolución de las herramientas inteligentes ha traído una mejora en los procesos de toma de decisiones, permitiendo el apoyo en experiencias obtenidas con anterioridad. El propio aprendizaje que se obtiene de estas se está incorporando cada vez más en distintos procesos empresariales y de toma de decisiones en

todo el mundo. Estos tienden a dejar información histórica que son utilizadas por estas herramientas inteligentes para dar soluciones a problemas nuevos que surjan con características similares.

Los sistemas basados en el conocimiento son técnicas de la inteligencia artificial que permiten desarrollar herramientas inteligentes que simulan el pensamiento de un experto. Estos sistemas permiten procesar información obtenida, tal es el caso de la experiencia en la gestión del tiempo. Esta experiencia parte de la información histórica de los proyectos que se desarrollan en la institución. La UCI cuenta con una estructura productiva organizada, además de varios años desarrollando software que posibilita la obtención de experiencia y de conocimientos sobre su propio proceso productivo.

En la UCI se utiliza para realizar la estimación un método basado en trabajos realizados por especialistas en la gestión de software de esta institución y parte de la experiencia que han obtenido estos especialistas en su desempeño profesional. Esta estimación se realiza en un único momento y se utiliza un solo método para hacerla, quedando imposibilitada la realización de comparaciones entre distintos enfoques brindados por los diversos métodos. Por otra parte se utiliza el mismo método para estimar proyectos con características totalmente distintas, en los que la presencia y el impacto de los indicadores definidos pueden tener comportamientos antitéticos y su traducción no estar en correspondencia con los resultados esperados.

El avance en los nuevos lenguajes de programación utilizados en el desarrollo de software, hacen que el tamaño del proyecto y el esfuerzo para realizarlo difiera considerablemente con respecto a lo que se obtenía hace algunos años atrás. A esto se le suma que las potentes herramientas en que se apoyan (Frameworks, IDE de desarrollo, hardware) mejoran todo el proceso de la programación. Esto no está comprendido dentro de los métodos de estimación de los proyectos de desarrollo de software, debido a que en su mayoría fueron elaborados antes del surgimiento de estas herramientas e intentan estimar de manera independiente a las tecnologías. Estos métodos no tienen como base el uso de la experiencia propia de la institución en función de esta estimación, desechando la información histórica de los proyectos anteriores que aportan un arsenal de elementos que apoyan la estimación.

Actualmente en la UCI no se recoge la experiencia y conocimiento de sus proyectos, teniendo en cuenta los procesos de la gestión del tiempo y todos los aspectos que influyen sobre la duración de este, incluyendo el uso de las nuevas tecnologías y metodologías de desarrollo. No se almacena toda esta

información para ser utilizada luego en próximas estimaciones de la duración de un proyecto de desarrollo de software.

Debido a esta problemática se plantea como **Problema a resolver:** ¿cómo utilizar la experiencia y el conocimiento en la estimación de tiempo de duración de los proyectos de desarrollo de software?

Definiéndose en la investigación como **Objeto de estudio** las formas de representar el conocimiento para estimar la duración de tiempo de los proyectos de desarrollo de software.

Para resolver el problema planteado con anterioridad, se trazó como **Objetivo general:** Desarrollar una base de casos para representar el conocimiento adquirido relacionado con la estimación de la duración de proyectos de desarrollo de software en la Universidad de las Ciencias Informáticas.

Siendo el **Campo de acción** la base de casos para representar elementos relacionados con la duración de proyectos de desarrollo de aplicaciones informáticas.

Quedando como **Objetivos Específicos:**

1. Definir indicadores relacionados con la estimación de la duración de proyectos de desarrollo de software.
2. Formalizar un modelo computacional donde se apliquen técnicas planteadas por los sistemas basados en conocimiento para representar la duración de proyectos de desarrollo de software.
3. Realizar análisis y diseño del sistema.
4. Implementar la solución propuesta.
5. Realizar las pruebas al sistema.

Se propone como **Idea a defender** el desarrollo de una base de casos para representar el conocimiento adquirido relacionado con la estimación de la duración de proyectos de desarrollo de software en la Universidad de las Ciencias Informáticas aprovecha la experiencia y el conocimiento para ser utilizado en estimaciones de tiempo de desarrollo en los proyectos.

Para lograr los objetivos trazados, se desarrollaron las siguientes **Tareas de la Investigación**:

1. Elaboración del marco teórico de la Investigación.

2. Caracterización de la situación existente en el mundo, en la región y en el país en particular sobre la estimación de la duración de proyectos de desarrollo de software, la inteligencia artificial y la técnica que determine utilizar para definir la posición de los investigadores.
3. Análisis de la técnica sistemas basados en conocimiento para aplicar sus componentes a la estimación de la duración de proyectos de desarrollo de software.
4. Identificación de los indicadores que permitan apoyar la determinación de tamaño y esfuerzo del desarrollo de un proyecto de software.
5. Validación de la propuesta de indicadores por expertos.
6. Definición de los requerimientos mínimos indispensables a ser resueltos con el sistema a desarrollar.
7. Obtención de los artefactos correspondientes al análisis.
8. Obtención de los artefactos correspondientes al diseño del sistema.
9. Obtención de los artefactos correspondientes a la implementación del sistema.
10. Definición de la estrategia, el tipo y los casos de prueba que posibiliten validar la solución propuesta.

**Métodos Científicos utilizados en la investigación.**

**Métodos teóricos.**

**Analítico–Sintético**: permitió el análisis y la síntesis de varios modelos de bases de casos así como métodos de estimación de tamaño y esfuerzo de proyectos de desarrollo de software contribuyendo a un mejor entendimiento de los mismos para definir la propuesta de los indicadores y realizar la estimación de tiempo y de la base de casos.

**Inductivo-Deductivo**: Se utilizó para el planteamiento del objetivo y la extracción de las ideas fundamentales para la elaboración y fundamentación del trabajo de diploma.

**Histórico-Lógico**: permitió el estudio de la evolución de la gestión del tiempo en los proyectos así como las distintas herramientas inteligentes para su tratamiento.

**Métodos empíricos.**

**Encuesta:** se realizó a un conjunto de expertos para realizar la validación de los indicadores propuestos.

## CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

### Introducción al capítulo.

En el presente capítulo se describen los antecedentes y el estado existente en el mundo, el país y en la Universidad de Ciencias Informáticas sobre la Gestión de Tiempo de un proyecto de desarrollo de software. Además se abordará sobre las técnicas de la inteligencia artificial en el proceso de apoyo al mismo. También se incluyen conceptos teóricos de las tecnologías a utilizar para implementar la solución de esta investigación, así como las metodologías de desarrollo empleadas en el proceso de solución.

### 1.1 Gestión del Tiempo.

El desarrollo de la Gestión del Tiempo de los proyectos comienza a partir de 1915 cuando Henry Laurence Gantt desarrollo y popularizó los gráficos de Gantt (8). Estos representan gráficamente el tiempo en forma de barras, muy útil para controlar el trabajo y registrar el avance de tareas en el tiempo definido. Su desventaja es que la interdependencia entre las diferentes actividades (la cual controla principalmente el progreso del proyecto) no puede determinarse a partir del diagrama de barras. Las complejidades crecientes de los proyectos exigieron técnicas de planeación más sistemáticas y efectivas, con el objetivo de optimizar la eficiencia en la ejecución del proyecto (9).

Para los proyectos grandes se requirió además el uso de técnicas basadas en redes de precedencia como: PERT (Program Evaluation and Review Technique o Técnica de Revisión y Evaluación de Programas) y CPM (Critical Path Method o Método de Ruta Crítica). El CPM fue desarrollado primero en 1957 por J. E Kelly de Remington y Rand y M.R Walker de DuPont como una aplicación a los proyectos de construcción y, posteriormente, se extendió a un estado más avanzado por Mauchly. PERT fue desarrollado en 1958 por Booz Allen y Hamilton, para la marina de los Estados Unidos de América, por una organización consultora, con el fin de programar las actividades de investigación y desarrollo para el programa de misiles Polaris. (9).

PMI (Instituto de Gestión de Proyecto del inglés “Project Management Institute”) publicó a principios de 1990 la primera edición del PMBOK (Project Management Body of Knowledge, Proyecto Órgano de Gestión del Conocimiento) la cual se convirtió en un pilar básico para la gestión y dirección de proyectos. PMBOK publicó su 4ta edición en el 2008.

Esta guía contiene y describe los conocimientos y habilidades necesarias dentro de la gestión de proyectos y está aprobado como estándar por el ANSI (American National Standard Institute, Instituto Nacional Americano de Estándares). PMBOK reconoce 5 grupos de procesos básicos: Inicio, Plan, Ejecución, Control y Monitoreo y Cierre y 9 áreas de conocimiento comunes a casi todos los proyectos: Gestión de la Integración, Gestión del Alcance, Gestión del Tiempo, Gestión de la Calidad, Gestión de Costos, Gestión del Riesgo, Gestión de Recursos Humanos, Gestión de la Comunicación, Gestión de las Compras y Adquisiciones. (6)

PMBOK permite describir y documentar proyectos de forma controlada y ordenada. Teniendo en cuenta esto, se utilizó esta guía en la descripción de los procesos relacionados con la estimación de la duración para dar solución al problema planteado.

### 1.1.1 Gestión del Tiempo de Proyecto de PMBOK.

La Gestión del Tiempo de Proyecto contempla los siguientes procesos (6):

- Definir las actividades: Es el proceso que consiste en identificar las acciones específicas a ser realizadas para elaborar los entregables del proyecto.
- Secuenciar las actividades: Es el proceso que consiste en identificar y documentar las interrelaciones entre las actividades del proyecto.
- Estimar los recursos de las actividades: Es el proceso que consiste en estimar el tipo y las cantidades de materiales, personas, equipos o suministros requeridos para ejecutar cada actividad.
- Estimar la duración de las actividades: Es el proceso que consiste en establecer aproximadamente la cantidad de períodos de trabajo necesarios para finalizar cada actividad con los recursos estimados.
- Desarrollar el cronograma: Es el proceso que consiste en analizar la secuencia de las actividades, su duración, los requisitos de recursos y las restricciones del cronograma para crear el cronograma del proyecto.
- Controlar el cronograma: Es el proceso por el que se da seguimiento al estado del proyecto para actualizar el avance del mismo y gestionar cambios a la línea base del cronograma.

El proceso de estimación de las actividades posee varias técnicas y herramientas que permiten realizar dicho proceso, entre ellas se encuentran: el Juicio de Expertos, Estimación Análoga, Estimación

Paramétrica, Estimación por Tres Valores y Análisis de Reserva. De estas, el Juicio de Expertos es guiado por la información histórica y puede proporcionar información sobre el estimado de la duración o las duraciones máximas recomendadas, procedentes de proyectos similares anteriores. (6)

La información histórica parte de un grupo de datos que debe ser almacenada para su análisis posterior por los expertos. Esta información parte de los proyectos pasados de la institución que realiza la estimación y deben reflejar en gran medida los indicadores del software que tienen un impacto en la duración final de este.

### 1.1.2 Herramientas de software de Gestión de Tiempo.

En la UCI, la gestión de proyecto utiliza varias herramientas que apoyan la gestión del tiempo. Estas herramientas se muestran a continuación:

**Microsoft Project**: Constituye una herramienta eficaz y flexible, la cual permite gestionar, administrar y realizar el seguimiento de todo tipo de proyectos, posibilitando la planificación de su ejecución, simplificando el trabajo y ofreciendo la posibilidad de realizar un seguimiento del trabajo de forma sencilla y cómoda. (10). Este presenta como inconveniente que es un software propietario. Además no proporciona una visión global de todos los proyectos que tiene una organización. Cada usuario debe tener una copia de la versión correcta de Microsoft Project instalado en su máquina, lo cual es costoso y toma mucho tiempo para instalar y mantener.

**Redmine**: es una aplicación web. Se trata de una herramienta multi-plataforma y de código abierto para la gestión de proyectos. Permite una visión general del proyecto así como su configuración, realizar peticiones que son las unidades de trabajo; pueden ser tareas, errores, o mejoras. Estas se asignan a personas y se puede ir siguiendo su evolución, tiempo dedicado y comentarios. Además muestra el avance del proyecto así como el porcentaje que queda para terminar un hito. (11) Esta herramienta está integrada a la producción en la universidad y se le han realizado varias adaptaciones que contribuyen a la gestión del proyecto, dentro de estos se encuentra GESPRO el cual constituye un paquete para la gestión de proyectos desarrollado por la UCI (12) y además permite calcular el tiempo de duración de un proyecto de desarrollo de software pero este cálculo no tienen en cuenta la experiencia continua que se va obtenido en el desarrollo de proyectos ni las ventajas que ofrece el desarrollo con las nuevas tecnologías utilizadas para el desarrollo.

**Trac**: es una herramienta de código abierto de gestión de proyectos y seguimiento de tareas. Trac permite hiper-enlazar información entre la base de datos de las tareas (tickets), las páginas wiki de contenido y el control de revisiones. También sirve como una sofisticada interfaz para el sistema de control de versiones. Su instalación es sencilla y está bien documentada y permite la inserción de nuevos informes en cada proyecto, seleccionando los criterios de concordancia. (13) A pesar de eso la herramienta no fue pensada para la gestión de proyectos en sí, sino desde el punto de vista del desarrollo de software. Cada proyecto es independiente y no se comparte la base de datos entre ellos y el sistema de permisos no es muy flexible. (13)

Estas herramientas son útiles en la gestión de proyecto y del tiempo, pues permiten la visión general del ciclo de vida del proyecto. Por otra parte gestionan la información previamente definida como la duración del tiempo, pero no influye en esta, debido a que no utilizan la experiencia obtenida con anterioridad como pueden ser los datos históricos de los proyectos que en la UCI se realizan.

Los datos históricos que permiten representar la experiencia y el conocimiento parten de las características de los proyectos que influyen sobre su duración. Estas características se definieron como indicadores de proyecto.

## 1.2 Indicadores de proyecto.

La Organización de Naciones Unidas (ONU) definen los indicadores como “Herramientas para clarificar y definir, de forma más precisa, objetivos e impactos (...) son medidas verificables de cambio o resultado (...) diseñadas para contar con un estándar contra el cual evaluar, estimar o demostrar el progreso (...) con respecto a metas establecidas, facilitan el reparto de insumos, produciendo (...) productos y alcanzando objetivos”. (14) Los indicadores deben ser medibles, la característica descrita debe ser cuantificable en términos ya sea del grado o frecuencia de la cantidad, ser reconocido fácilmente por todos aquellos que lo usan y deben ser controlable dentro de la estructura de la organización. (3)

El uso de los indicadores provee de una herramienta que posibilita obtener los aspectos que tengan un impacto en la duración de un proyecto de desarrollo de software. La representación de estos indicadores refleja la información referente a la duración que se obtiene del desarrollo de un proyecto. Se realizó un estudio de los métodos de estimación que posibiliten la obtención de esta información.

### 1.2.1 Métodos de estimación de tiempo.

Para la selección de indicadores que influyen en el tiempo, se tiene en cuenta los métodos de estimación de tamaño y esfuerzo de un proyecto, debido a la incidencia de estos factores sobre la duración del mismo. Teniendo esto en cuenta se realizó un análisis de algunos de los métodos más utilizados y mejor referenciados de la Universidad de la Ciencias Informáticas, de Cuba y del Mundo sobre la estimación de tiempo, tamaño y esfuerzo de proyectos de desarrollo de software. Dichos métodos se muestran a continuación:

**Puntos de Caso de Uso.**

Este método de estimación de proyectos de software fue desarrollado en 1993 por Gustav Karner de Rational Software y está basado en una metodología orientada a objetos. Surgió como una mejora al método de puntos de función pero basando las estimaciones en el modelo de casos de uso, producto del análisis de requisitos (15). Su actor considera la funcionalidad vista por el usuario como la base para estimar el tamaño del software. El objetivo de la técnica es estimar las horas necesarias para ejecutar un conjunto de casos de uso. Para ello, cuantifica la complejidad del sistema y el tiempo necesario para producir una unidad de complejidad.

Entre los problemas que presenta este método está la inexistencia de un estándar para describir casos de uso, eso hace que aplicar la métrica sea más engorroso. No se distingue un criterio único aplicado para el cálculo de la complejidad de un caso de uso. Además, también existen diferentes criterios para identificar una transacción. (16)

**Puntos de Función.**

El modelo Puntos de Función (PF) de Alam Albrecht fue publicado en 1979. Su propósito es medir el software cuantificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema. Los objetivos de los Puntos de Función son: medir lo que el usuario pide y lo que el usuario recibe independientemente de la tecnología utilizada en la implantación del sistema, proporcionar una métrica de tamaño que dé soporte al análisis de la calidad y la productividad, proporcionar un medio para la estimación del software y proporcionar un factor de normalización para la comparación de distintos software. (17)

Este método carece de precisión cuando se trata de proyectos pequeños. Resulta arduo formar al personal en su utilización y más todavía mantener unos criterios homogéneos de recuento. Requiere dedicación adicional en los proyectos que tienen presupuesto ajustado. El factor de ajuste calculado a partir de las características generales del sistema resulta de dudosa utilidad. (18)

**COCOMO.**

En el año 1981 Barry Bohem publica el modelo COCOMO, acorde a las prácticas de desarrollo de software de aquel momento. Durante la década de los 80, el modelo se continuó perfeccionando y consolidando, siendo el modelo de estimación de costos más ampliamente utilizado en el mundo. (19) Con la evolución de las tecnologías cambia la forma de desarrollar y se crea una nueva versión atendiendo a ellas, el modelo a COCOMO II.

Este modelo presenta como inconvenientes que el número de líneas de código en el modelo, no es un factor representativo ya que la diferencia entre lenguajes desarrollados en la actualidad puede diferir de uno a otro. En los lenguajes orientados a objetos, existe el concepto de encapsulamiento, el cual traduce el código neto de la aplicación, a un mínimo de líneas de trabajo.

**PROBE.**

PROBE (PROxy Based Estimating) por sus siglas en inglés, traducido al español se entiende como Estimación basada en la evaluación. El método PROBE descrito por Humphrey defiende el uso de medidas seleccionadas personalmente y modelos de regresión para estimar el tamaño de un producto de software. Un proxy es una medida de tamaño que puede ser utilizada para estimar la longitud de un proyecto medido en líneas de código. Se utilizan modelos de regresión simples para estimar las líneas de código desde las medidas de tamaño basadas en proxys y las horas de trabajo desde las líneas de código. (20)

**Método de estimación UCI.**

El desarrollo del método de estimación utilizado en la UCI, partió de la necesidad de estimar el tamaño, costo y esfuerzo requerido para desarrollar un producto de software en esta institución. Esto se debió principalmente a que no contaba con una base histórica que permitiera realizar la estimación basada en la experiencia.

El trabajo de Dayami Rodríguez (21) hace un recuento de su desarrollo y describe cada paso. En este se expone el método de estimación, las métricas que conforman la base y el proceso de retroalimentación organizacional basado en indicadores de estimación.

Estos métodos permiten tener en cuenta variables de la institución ajustando sus resultados a cada empresa que realice la estimación con ellos. La dificultad recae cuando se desea tener en cuenta criterios de otros métodos, ya que se pueden tener diferentes visiones de los indicadores sobre un mismo proyecto y todos influyen en este. Por lo que se propone la integración de estos métodos para definir indicadores generales que tengan en cuenta la mayor cantidad de características que distinguen a un proyecto. Además se propone la integración de otros indicadores que tengan en cuenta el uso de las nuevas tecnologías y características propias de la UCI que afecten la duración de un proyecto en la institución.

La ciencia de la computación trata cada vez de resolver nuevos problemas que sean difíciles de solucionar mediante las técnicas computacionales existentes. Las decisiones que normalmente se presentan en la vida empresarial pueden caer en esta clase de problemas. (22) En el ámbito internacional se han desarrollado diversas técnicas de la inteligencia artificial que permiten su aplicación en los procesos empresariales y son ampliamente utilizadas en funciones que realiza el hombre sobre un área de conocimiento, por su capacidad de procesamiento y rapidez de respuesta.

### 1.3 Inteligencia artificial.

La inteligencia artificial (IA) es un término difícil de plantear aunque es ampliamente utilizado. Una definición generalmente aceptada en los ámbitos académicos, es la propuesta por Barr y Feigenbaum en la cual se destaca que son sistemas inteligentes que tratan de efectuar tareas que normalmente se asocian con el comportamiento humano inteligente, tales como comprensión del lenguaje, aprendizaje, razonamiento, solución de problemas y otras similares. (23)

La IA convencional basada en el análisis formal y estadístico del comportamiento humano ante diferentes problemas, hace un uso de datos y conocimientos previos para simular el comportamiento humano o un comportamiento lógico, obteniendo a su vez experiencia que incorpora al sistema. Sus diversos métodos posibilitan la obtención de soluciones a problemas de análisis de situaciones (24). De estos métodos se derivan una serie de sistemas basados en el conocimiento que, en gran medida, permiten simular las funciones de expertos humanos.

### 1.3.1 Sistemas Basados en Conocimiento.

Se define como Sistemas Expertos o Sistemas Basados en el Conocimiento (SBC) a “los sistemas de computación que recopilan y simulan el pensamiento y la experiencia de expertos humanos en un área específica del conocimiento”. (25) Estos sistemas son capaces de procesar y memorizar información, aprender y razonar en situaciones determinísticas e inciertas, comunicarse con humanos, sistemas expertos y/o sistemas en general, tomar decisiones apropiadas y explicar el por qué estas decisiones han sido tomadas. De esta forma, los SBC actúan como un consultor que puede proporcionar ayuda a un experto humano con un grado razonable de credibilidad.

Los SBC están formados por tres componentes principales: la base de conocimiento, la máquina de inferencia y una interfaz de usuario.

**Base de conocimiento**: Contiene el conocimiento y las experiencias de los expertos en un determinado dominio representado por medio de símbolos. (25)

**Motor de Inferencia**: Es la parte del sistema experto 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. (25)

**Interfaz de usuario**: Es la parte del sistema experto que permite la comunicación entre el usuario y el motor de inferencias. Adicionalmente, permite introducir la información que necesita el sistema y comunica las respuestas del sistema experto al usuario. (25)

El conocimiento se almacena en la base de conocimientos. Este se puede representar mediante diversas formas de representación del conocimiento que modelan cómo se relacionan los valores de los diferentes rasgos que caracterizan el dominio, pesos de una red neuronal, casos o ejemplos de problemas del dominio. (26) Esta representación del conocimiento ha dado lugar a la especialidad conocida como “Ingeniería de conocimiento” con el fin de llevar ideas, conceptos y experiencia de los seres humanos a su representación en casos para poder inferir sobre este.

### 1.3.2 Ingeniería del conocimiento.

El trabajo de Yenier F. Moreno (27) refiere a la definición de ingeniería de conocimiento por Feigenbaum y McCorduck como: “El arte de conducir los principios y herramientas de la IA para tener aplicaciones de

problemas difíciles que requieren el conocimiento del experto para su solución. Los aspectos técnicos para adquirir este conocimiento, representarlo y usarlo apropiadamente para construir y explicar líneas de razonamiento son problemas importantes en el diseño de los sistemas basados en conocimiento."

La ingeniería de conocimiento permite la adquisición, representación, validación, explicación del conocimiento y está presente en el desarrollo de los SBC. Esta permite codificar y hacer explícitas la representación del conocimiento que usan los expertos humanos en la resolución de problemas reales.

En la ingeniería del conocimiento se identifican dos actividades relacionada con la manera en que el conocimiento se estructura mediante una forma de representación, estas son:

Adquisición del conocimiento: Se refiere a la extracción del conocimiento de los expertos humanos, libros, documentos, sensores, archivos de computadora, entre otros (27) permitiendo obtener la información que se quiere representar.

Representación del conocimiento: Como parte de esta actividad, el conocimiento adquirido es organizado y codificado en la base de conocimiento. (27) De esta forma queda conformada la información que se va a almacenar en la base de conocimiento.

Estas actividades permiten definir qué conocimiento es el relevante y cómo se va a representar. Se tiene en cuenta además la naturaleza, la emisión de la información y la certidumbre que puede tener el conocimiento, de esto dependerá la representación.

El estudio realizado a los métodos de estimación y los documentos referidos a la estimación en la UCI permitió obtener el conocimiento que se desea representar. La naturaleza de la información que manejan los métodos es descriptiva, con un conjunto de datos provenientes de observaciones directas sobre el producto a estimar, conformado por un grupo de características representada por el par atributo-valor. La emisión de la información es realizada mediante fuentes proveniente de la institución, mediante observación, análisis o interrogación. La certidumbre se manifiesta de forma incierta en la emisión del conocimiento, siendo esta siempre positiva, ya que la estimación se realiza con valores conocidos.

### 1.3.3 Formas de representación del conocimiento.

Existe un gran número de representaciones del conocimiento. Cada una intenta representarlo como se muestra en la vida real. Obtenidas las principales características del conocimiento que se desea

representar a continuación se detalla el proceso de selección de la representación de este, apoyado en diversas bibliografías se realizó un estudio de los más referenciados y utilizados:

El **cálculo de predicado** representa una extensión de la lógica proposicional. Sus elementos fundamentales son el objeto y el predicado (26). Esta representación conceptual posibilita relacionar los distintos predicados almacenados y son más utilizados en casos donde se puedan tener varias situaciones que se necesiten relacionar, en el caso que trata la investigación, no se modela el conocimiento en base a relaciones conceptuales entre estos.

Las representaciones por **frames** se desarrollaron con el propósito de superar las limitaciones de otros modelos, como las redes asociativas, las reglas y la lógica, para agrupar información y establecer relaciones entre esos grupos. (28) El Dr. Daniel Gálves recomienda su utilización donde sea necesario tener descripciones estructurales complejas para describir adecuadamente el dominio de la aplicación. Los frames son buenos para almacenar el conocimiento sobre los elementos de un dominio de aplicación que tiene una descripción estereotipada, no siendo así con el conocimiento que se trabaja ya que este se describe como un grupo de objetos descritos por características de forma lineal.

Las representaciones mediante **listas de soportes** y **conocimiento difuso** se basan en la veracidad o el mínimo conocimiento de un hecho para realizar una deducción. (26) Esta forma de representación cumple objetivo cuando es incierta en gran medida la información o existe conocimiento difuso de la información.

Otras formas conocimiento permiten realizar una representación de objetos. Las **redes semánticas**, son descripciones simbólicas que constan de objetos y relaciones, ilustrándose en forma de red. (26) Lo que impide la independencia entre cada uno de los objetos.

Las **reglas de producción** son muy semejantes al cálculo de predicados (26). Posee un grupo de ventajas que la hacen ser una de las más usadas, entre estas se encuentran su modularidad: cada regla es una unidad de conocimiento que puede ser añadida, modificada o removida independientemente de las otras reglas existentes, permitiéndole al sistema gran flexibilidad (28); uniformidad: todo el conocimiento del sistema se expresa en el mismo formato (28); naturalidad: las reglas son un formato natural para expresar conocimiento en algunos. (28)

También posee dificultades y desventajas que impiden su uso en otras áreas de conocimiento como son:

- Encadenamiento infinito: se produce debido a que la mayoría de los sistemas basados en reglas realizan una búsqueda primero a profundidad y este método padece este problema si no es implementado correctamente (28)
- La ineficiencia: el reconocimiento de qué reglas son aplicables en cada ciclo es altamente ineficiente; durante cada ciclo el motor de inferencia examina cada regla para encontrar la aplicable. Se han desarrollado algunas técnicas para tratar este problema, pero ninguna da una solución completa. (28);
- Cubrimiento del dominio: Hay dominios en que las entradas varían mucho y requerirán miles de reglas para considerar todas las situaciones. (28) Esta es considerada una de las grandes desventajas de las reglas, que imposibilita su uso en la representación del conocimiento que trata esta investigación, debido a que el número de características que se debe representar por cada objeto conllevaría a un gran número de reglas a ser diseñadas por un experto haciendo este, un proceso muy engorroso.

La representación mediante **casos** permite tener en cuenta las características que presenta el conocimiento que aborda la estimación de proyecto. Esta representa el conocimiento como un objeto con un grupo de características, permitiendo representar cada ocurrencia de un proyecto como se describe mediante los métodos de estimación y la bibliografía referente a la estimación y gestión del tiempo.

Como ventajas el Dr. Daniel Gálves (28) presenta las siguientes:

- Experiencias previas que hayan sido exitosas pueden ser utilizadas para justificar nuevas soluciones.
- Experiencias previas que no hayan sido exitosas se pueden utilizar para anticipar problemas.
- La comunicación entre el sistema y los expertos se realiza en base a ejemplos concretos, el sistema explica sus decisiones citando precedentes.
- Los sistemas de representación de casos permiten proponer soluciones para los problemas rápidamente, evitando el tiempo necesario para derivar respuestas desde el estado inicial de un proceso de búsqueda de soluciones. Esta ventaja se manifiesta principalmente en situaciones donde un sistema basado en reglas, hubiese requerido realizar una larga cadena de inferencias para alcanzar una solución.
- Los casos ayudan a focalizar el razonamiento sobre las partes importantes de un problema señalando que rasgos del problema son importantes.
- Es aplicable a un amplio rango de problemas.

De esta forma se define la representación mediante casos como forma de representación de conocimiento.

### 1.3.4 Sistema de Razonamiento Basado en Casos.

El funcionamiento de un sistema de Razonamiento Basado en Casos (RBC) es un enfoque que aborda nuevos problemas tomando como referencia problemas similares resueltos en el pasado. De modo que problemas similares tienen soluciones similares, y la similitud juega un rol esencial (29).

La figura 1 muestra las cuatro etapas de los sistemas de RBC (30):

1. Recuperar el caso (o casos) más similar al problema planteado.
2. Reutilizar el caso (o casos) para intentar resolver el problema.
3. Revisar la solución propuesta.
4. Retener la nueva solución para que forme parte de la base de casos.

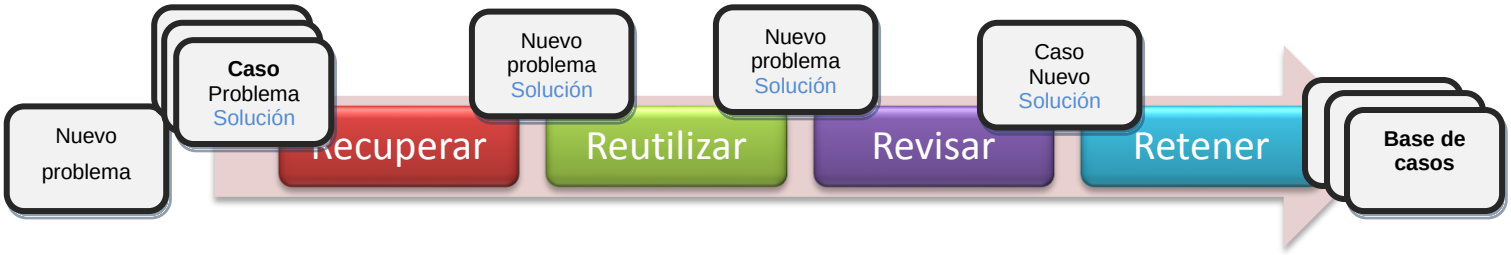


Figura 1: Etapas de los sistemas de RBC.

El sistema funciona fundamentalmente con una base de casos y un solucionador de problemas. La base de casos contiene las descripciones de los problemas resueltos previamente. Cada caso puede describir un episodio particular o una generalización de un conjunto de episodios relacionados. En el estilo de solución de problemas se recupera un caso semejante al nuevo y la solución del problema recuperado se propone como solución potencial del nuevo problema. Esto se deriva de un proceso de adaptación en el cual se adecua la vieja solución a la nueva situación.

## 1.4 Base de Casos.

El automatizar los sistemas de RBC lleva implícito el problema del almacenamiento de las experiencias o casos. El caso representa situaciones experimentadas previamente y se distingue de otras formas de conocimiento o experiencia porque ellos son ejemplos específicos de hechos. El caso debe contener la

descripción del problema o situación; una solución, es decir información que no pueda ser derivada a partir del nivel actual de conocimiento y por último, información acerca de si la solución fue correcta o incorrecta. (31)

Para almacenar los casos debe usarse una memoria o Base de Casos (BC) que posea propiedades similares a la memoria humana, tales como (31):

- Ser ilimitada.
- En la medida en que la memoria crezca no puede hacerse más lenta.
- Debe permitir buscar directamente los elementos de memoria que sean relevantes para un problema.

### 1.4.1 Descripción de los casos.

Los autores coinciden con los estudios realizados por Kotony Chase (32) donde consideran los casos como una colección de rasgos (un par atributo-valor) los cuales están organizados en una generalización, mostrándose en la forma en que se describen las características de un proyecto: Cantidad de PC, 20; Capacidad del analista: Alto; Buena comunicación entre los miembros del equipo: Considerable o Duración, 20 meses. Por lo que, los casos están descritos a partir de los valores que se le asignan a los rasgos predictores y a los rasgos objetivos. Los rasgos predictores son los que determinan los valores de los rasgos objetivos.

Cada base de casos está formada por casos que pertenecen a un área de dominio específico, por lo que todas son diferentes de acuerdo a la complejidad del área con que se relacionen. Además la selección del conjunto de rasgos por el cual se conformarán los casos es una cuestión central debido a que determina qué información será almacenada en memoria por cada uno de sus elementos dada la descripción de un nuevo problema.

**Estructura de las bases de casos.**

Debido a que los casos constituyen el elemento principal de todo RBC, la manera en que se almacenan en la memoria repercutirá directamente en el rendimiento de este. De esta forma la BC es la encargada de almacenar y organizar los casos disponibles y su estructura es crucial para la fase de recuperación de casos similares. (33)

Existen gran cantidad de variantes en la forma que se organizan las BC. Estas variantes se pueden agrupar en dos categorías principales: la plana y la jerárquica:

**Memoria plana**: En esta estructura se presentan los casos completos de forma secuencial utilizando algún tipo de indexación ya sea manual o automática. Esta estructura tiene como desventaja que la búsqueda en la misma es costosa computacionalmente, por lo que es recomendable para BC pequeñas. 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 caso respetando el método de indexación que se esté utilizando. (33)

**Memoria 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 a los mismos. Esta estructura sacrifica sencillez en la inserción de los nuevos casos por eficiencia en los algoritmos de búsqueda. (33)

Se debe realizar un consenso entre la eficiencia de recuperación y eficiencia de inserción para realizar la selección de una de las estructuras anteriores para crear la base de casos. De la correcta elección de la estructura dependerá en gran medida la eficiencia global del funcionamiento del sistema.

Para seleccionar la estructura de la memoria que modelará la BC, se tuvo en cuenta la gran cantidad de información que puede definir un proyecto. La estimación se realiza con todos los elementos necesarios por lo que no requiere de estructuras que modelen entradas de datos incompletos, además no existe una relación conceptual entre los proyectos, cada proyecto se estructura por sus características, valores e incertidumbre de forma independiente. Debido a esto, se seleccionó la estructura de la memoria plana para la BC.

La memoria plana, almacena la información en una matriz. Esta estructura tiene como ventaja que la inserción de los casos en la memoria es poco costoso y a diferencia de las memorias jerárquicas no almacena datos auxiliares para realizar la búsqueda minimizando el espacio que ocupan los casos (28). Además el procesamiento actual de las tecnologías permite que el crecimiento de la base de casos no se vea afectado en el rendimiento de la aplicación.

### 1.4.2 Ejemplos de bases de casos.

La representación y manejo de las bases de casos requiere de un gran esfuerzo. La mayor parte de los trabajos encontrados en las revisiones bibliográficas sobre representación de los casos, están relacionados a aplicaciones particulares, y esto se debe a que la representación de los casos siempre depende de la tarea y el dominio para el cual se diseñe el RBC.

El trabajo del Dr. Daniel Gálves (28) muestra el desarrollo de varios enfoques para la representación de los casos que son usados ampliamente en la creación de muchas de las BC:

- El Case-based Search System utiliza casos que contienen soluciones completas de problemas, cada caso es una lista de estados de problema desde un estado inicial a un estado objetivo. Cada caso implícitamente contiene muchos otros como subcasos.
- En el sistema CHEF representa los casos como planes.
- El sistema GREBE, usa un lenguaje de representación en el cual relaciones intencionales, temporales y causales arbitrariamente se podrían enunciar explícitamente en un dominio legal mediante proposiciones.
- En SQUAD, se utilizan registros de una base de datos relacional como casos.

Estas representaciones solo permiten su uso para el área de conocimiento que fueron desarrolladas. Por otra parte brindan una visión de cómo se puede modelar una BC dependiendo de su dominio.

A nivel nacional hay varios centros que se destacan por los resultados que tienen en las investigaciones sobre este campo. Entre estos se encuentran:

- La Universidad de las Villas. Esta institución cuenta con una gran cantidad de estudios realizados sobre los SBC (22) (34), se destacan varios trabajos realizados sobre Sistemas Enseñanza/Aprendizaje Inteligentes (STI) (29) que aplican a varias ramas de la enseñanza a distintos niveles.
- El Centro de Aplicaciones de Tecnologías de Avanzadas (CENATAV) publica en su página web (35) más de 150 trabajos referentes principalmente al reconocimiento de patrones, minería de datos e ingeniería en sistemas.

En la UCI se han realizados varios trabajos sobre BC, entre ellos se pueden citar:

- Base de Conocimiento para inferir el comportamiento de las pruebas de liberación en el Laboratorio Industrial de Pruebas de Software: abarca el desarrollo de una base de conocimiento que permita inferir el comportamiento de las pruebas de liberación en el LIPS. (27)
- Modelación de una Base de Conocimientos. Aplicación de la Teoría de Errores: La misma se basa en el establecimiento de un modelo que permita la medición cuantitativa del conocimiento, al que luego se aplica un análisis de la teoría de errores. (36)
- Propuesta de base de conocimientos de las actividades del piloto del Sistema Integral de Gestión Cedrux. Se desarrolló con el fin de mejorar la toma de decisiones en las actividades de implantación del grupo Implantación y Soporte del CEIGE. (37)

Estas bases de conocimiento no pueden ser utilizadas para estimar la duración de proyecto. Todos los SBC no son iguales, y no basan su resolución en el mismo principio. Las BC también cumplen con estas características, debido a que el conocimiento es dependiente del dominio del área donde el sistema es protagonista.

En la bibliografía consultada, los autores de la presente investigación no encontraron referencias de una base de casos con el fin de almacenar la experiencia y el conocimiento en la estimación de tiempo de proyectos. Por esto consideran como solución el desarrollo de una BC para estimar la duración de proyectos de desarrollo de software que permita dar solución al objetivo de la investigación.

## 1.5 Metodología de desarrollo de software.

Una metodología de desarrollo de software es un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevos software. (38). Existen varias metodologías para diferentes tipos de proyectos. Las metodologías ágiles se centran más en el producto final, las mismas se encargan de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados, por lo que se hace más importante crear un producto software que funcione que escribir mucha documentación. (39)

En el presente trabajo de diploma se utilizará una metodología ágil para realizar el desarrollo del software, teniendo como premisa principal el artefacto funcional. Además, el equipo de desarrollo está integrado por

dos personas y se dispone de poco tiempo para su realización. Se seleccionó de las metodología eXtreme Programming (XP).

### 1.5.1 Programación Extrema.

La metodología de desarrollo XP se basa en la simplicidad, la comunicación y el reciclado continuo de código, se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP tiene como punto de partida la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y facilidades para enfrentar los cambios. (40)

La selección de esta metodología de desarrollo de software se basa en las ventajas que tiene para este trabajo diploma. Permite una implementación organizada contribuyendo a una menor taza de errores, debido que el cliente al estar incluido como parte del equipo de desarrollo monitorea constantemente el proyecto. Promueve el trabajo en equipo y da facilidades en cuanto a la cantidad de documentación que se genera en relación a las metodologías pesadas, lo que posibilita centrarse en el propio desarrollo del sistema. Además es recomendable emplearla en proyectos a corto plazo como el que incluye en esta propuesta de solución.

## 1.6 Herramientas de desarrollo.

### 1.6.1 Herramientas 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 (41). Entre las propiedades que tiene este lenguaje se encuentra que modela estructuras complejas, soporta estructuras orientadas a objetos, como objetos, clases y componentes.

Visual Paradigm para UML es una herramienta que presenta disponibilidad en múltiples plataformas, se integra en los principales entornos de desarrollo integrado. Utiliza un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación entre ellos. Provee soporte para la generación de código

y la ingeniería inversa. (42). Se utiliza esta herramienta debido a que permite construir los diagramas utilizado para representar las clases y el modelo de la base de datos. Se utilizó la versión 8.0 del Visual Paradigm para UML.

### 1.6.2 Lenguaje de programación. Groovy.

Se entiende como lenguaje de programación aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; mediante el cual se le da instrucciones al ordenador. Groovy es un lenguaje orientado a objetos para la Plataforma Java, como alternativa al lenguaje de programación Java. Groovy es un lenguaje ágil y dinámico para la Máquina Virtual de Java. Se basa en los puntos fuertes de Java, pero tiene características adicionales de potencia inspirado en lenguajes como Python, Ruby y Smalltalk. (43) Permite utilizar las funciones de programación disponibles para los desarrolladores de Java, el código se convierte en fácil de leer y mantener, debido a la sencillez de su sintaxis. Se integra perfectamente con todas las clases existentes en las bibliotecas de Java y compila directamente al bytecode de Java. Se utilizará la versión de Groovy 1.7.9.

### 1.6.3 Plataforma de desarrollo. Java.

Una plataforma de desarrollo es un entorno común donde se desenvuelve la programación de un grupo de aplicaciones. La plataforma Java es un entorno capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a bytecode. Provee una máquina virtual que se encarga de la ejecución de aplicaciones, un conjunto de bibliotecas estándares que ofrecen funcionalidades comunes así como una interfaz para la programación de aplicaciones haciendo uso del lenguaje Java. Está compuesta por una gran cantidad de tecnologías como Java Edición Estándar (JSE), Java Edición Micro (JME) y Java Edición Empresarial (JEE). (24) Esto permite interactuar con la máquina virtual de Java y explotar el conjunto de funcionalidades brindadas por la misma así como sus bibliotecas. Además, las aplicaciones sobre esta plataforma pueden también ser desarrolladas en la web. Para el desarrollo de aplicaciones, se utiliza un conjunto de herramientas conocidas como JDK (Java Development Kit, o herramientas de desarrollo para Java). Su utilizará la versión 1.7 de la herramienta JDK.

### 1.6.4 Framework. Grails.

El framework Grails de desarrollo web es un sistema de código abierto que aprovecha el lenguaje Groovy y complementa el desarrollo web en Java. Grails está licenciado bajo la licencia Apache Software liberal 2.0. Presenta un ciclo de desarrollo muy ágil que permite centrarte en las funcionalidades en lugar de los requisitos técnicos del framework. (44)

Grails es un framework que provee de un gran número de mecanismos para el desarrollo web a través de la tecnología base y sus complementos asociados. Incluye características como: reutiliza tecnologías Java ya probadas como Hibernate y Spring bajo una interfaz simple y consistente, ofrece documentación para las partes del framework relevantes para sus usuarios, bibliotecas de etiquetas dinámicas para crear fácilmente componentes web y proporciona un entorno de desarrollo orientado a pruebas.

Presenta además características que intentan incrementar su productividad en la plataforma Java:

- El modo de desarrollo automáticamente recarga los cambios realizados al código de la aplicación sin tener que reiniciar el servidor web.
- Soporte de tecnologías para la implementación de vistas de datos utilizando “dynamic tag libraries (tags)” y “groovy server pages (gsp)”.
- Funcionalidad disponible mediante métodos dinámicos. Posibilita utilizar métodos que se crean en tiempo de ejecución y sin estar implementados previamente.
- Ofrece mapeo de persistencia automático para las clases del dominio, así como también manejo automático de las relaciones entre las entidades.

Se utilizará la versión de 1.3.7 de Grails.

### 1.6.5 Entorno de Desarrollo Integrado. IntelliJ IDEA.

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés Integrated Developtment Enviroment) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un lenguaje de programación o varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. (45)

IntelliJ IDEA es un entorno para desarrollar aplicaciones Java, cliente y servidor. Posee un avanzado editor de código, compatible con las tecnologías utilizadas en este trabajo y dentro de un mismo entorno, permite análisis del código, análisis de dependencias y soporte para plug-ins. Este IDE permite a los

programadores centrarse en el desarrollo mientras que ejecuta funciones de configuración y salva de datos. Además provee de una consola interna para ejecutar comandos en el proyecto. Se utilizará la versión 11.0 del Intellij IDEA.

### 1.6.6 PostgreSQL.

PostgreSQL es un avanzado sistema de bases de datos relacionales basado en código abierto. Se encuentra disponible en casi todos los sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows. Presenta documentación muy bien organizada, pública y libre. Es altamente adaptable a las necesidades del cliente. También soporta todas las características de una base de datos profesional, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas. (46)

Es necesario para el presente trabajo, la utilización de un gestor de base de datos potente para el almacenamiento de un alto volumen de información y que sea libre para no incurrir en violaciones derivadas del uso de herramientas privativas y así contribuir a alcance de la soberanía tecnológica. Por lo que fue seleccionado PostgreSQL por ser software libre y ser un sistema estable, de alto rendimiento y gran flexibilidad, siendo estos requisitos indispensables para un sistema que puede albergar gran cantidad de información con un alto nivel de integridad y consultado constantemente. Se utilizará la versión PostgreSQL 8.4.

### 1.6.7 CSS.

Las Hojas de Estilo en Cascada o CSS (Cascading Style Sheets) son mecanismos simples que posibilitan aplicar formato a los documentos escritos en HTML, dando la posibilidad de separar el contenido de la presentación. (47) Describe cómo se verá un documento en pantalla, permitiendo darle formato a los documentos así como las múltiples páginas web al mismo tiempo. Esto permite aplicar el mismo estilo a diversas vistas que se muestren de la aplicación.

### 1.6.8 Sistema de control de versiones.

Subversion es un sistema de control de versiones libre y de código fuente abierto. Es decir, que maneja ficheros y directorios a través del tiempo. Permite acceder al repositorio a través de redes, lo que posibilita que sea usado por personas que se encuentran en distintos ordenadores. (48) Esta herramienta le permite

al equipo de trabajo conformado por dos personas trabajar sobre una misma aplicación, así como poder recuperar el trabajo realizado con anterioridad al llevar el control de cada una de las versiones anteriores.

**Conclusiones.**

En el presente capítulo se realizó un estudio del estado del arte del proceso de gestión del tiempo y los sistemas que sobre esta área se desarrollan, así como las herramientas inteligentes que permiten la simulación de expertos mediante el análisis de información histórica. Este estudio arrojó como resultado que los sistemas estudiados no satisfacen las necesidades de la UCI en cuanto al almacenamiento de la experiencia. Se definió que la representación del conocimiento se obtenga mediante la selección de indicadores y esta se represente por casos, para lo que se concluyó que la solución adecuada es el desarrollo de una base de casos que permita representar este conocimiento para ser usado en la estimación de proyectos de desarrollo de software. Se seleccionó XP como metodología para guiar el proceso de desarrollo de software. El sistema se desarrolla sobre el framework Grails en su versión 1.3.7 que utiliza a la plataforma Java y el lenguaje de programación Groovy versión 1.7.9. El IDE de desarrollo utilizado es el Intellij IDEA 11.0. Se seleccionó PostgreSQL 8.4 como sistema gestor de base de datos y el Subversion como el controlador de versiones.

## CAPÍTULO 2: DISEÑO DE LA BASE DE CASOS

### Introducción al capítulo.

En el presente capítulo se especifican los componentes de la base de caso. Se define el modelo que representa la memoria de la base de casos así como los factores que pueden influir en los datos que se almacenan. Se realiza la selección de los indicadores que influyen en la estimación de la duración de tiempo de un proyecto de desarrollo de software. Para esto se realizó un análisis de los métodos de estimación seleccionados por los autores que puedan aportar datos y ser usados para estimar el tamaño y el esfuerzo a realizar para concluir en un determinado tiempo un proyecto. Posteriormente se validó con expertos la selección de los indicadores propuestos. Se describe la composición de los rasgos y la fase de recuperación de los casos.

### 2.1 Definición del modelo de la Base de Casos.

La creación de la BC tiene en cuenta los siguientes aspectos: 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. (49) Estas características permiten estructurar la BC y cada uno de sus componentes, e influyen en la rapidez con que se puedan obtener y almacenar los casos.

Para definir la BC se tiene en cuenta el módulo encargado de realizar la búsqueda en la memoria donde están almacenados los casos semejantes a la entrada definida. El objetivo de este módulo es, dado una descripción, encontrar o inferir los valores para el resto de los casos de interés para lo cual es necesario encontrar los elementos de la memoria que sean semejantes al problema, según su descripción (49). Se usará el término patrón de búsqueda para referirse a la descripción del nuevo problema. La búsqueda se realiza a partir del patrón de búsqueda y se selecciona un conjunto de elementos de memoria candidatos. A este proceso se le llama recuperación.

En la recuperación se buscan y recuperan casos candidatos, es decir, aquellos que potencialmente son los más apropiados para hacer predicciones relevantes sobre el nuevo problema. En la etapa de selección se determina cuál o cuáles de los casos recuperados es el más similar al patrón de búsqueda mediante un algoritmo definido para realizar esta función (49). La BC debe ser capaz de permitir la búsqueda de cada uno de los casos contenidos en ella.

En la etapa de recuperación se tiene en cuenta además, otras características que influyen en la semejanza de los casos.

En el proceso de recuperación ha de tenerse en cuenta el manejo de la incertidumbre, el análisis de la función de semejanza entre rasgos, así como el umbral de semejanza entre los mismos. En esta investigación la incertidumbre precisa ser valorada tanto a nivel específico de los rasgos, como a nivel de casos. A continuación se dan elementos generales de cada uno de ellos.

**La incertidumbre.**

El trabajo realizado por Gutiérrez, Bello y Tellería, refieren que “La información incompleta es omnipresente: casi toda la información que se tiene del mundo es no cierta, completa o precisa”. (50) La incertidumbre es parte del mundo real, por lo que los sistemas RBC deben ser capaces de modelar esta realidad.

Desde la perspectiva del enfoque basado en casos los problemas de decisión se describen de la forma siguiente: los casos en la base se componen de un conjunto de rasgos predictores de la forma (valor, incertidumbre) y un rasgo objetivo de la forma (valor, incertidumbre). (51)

Esta incertidumbre viene dada por la veracidad con que se afirma que un rasgo se comporte de cierta forma. Cuando se realiza la estimación del tiempo se manifiesta en el grado de seguridad con que se presenta una característica en un proyecto. Esto influye en la similitud que pueda tener un rasgo de un caso con respecto a otros.

**Funciones de semejanza.**

La función de semejanza es la encargada de determinar el grado de semejanza que tiene el caso a resolver con respecto a cada uno de los casos contenido en la BC. No existe una medida de semejanza única o general para cualquier dominio, de ahí que la eficiencia del sistema radica en la función de semejanza que se defina. (29)

En la búsqueda por semejanza, el objetivo es recuperar el elemento de la memoria que más se parezca o asemeje al nuevo problema a partir del cálculo de una medida que cuantifique ese grado de similitud.

**Umbral de semejanza.**

El umbral de semejanza define en el proceso de recuperación el mínimo valor de similitud que debe tener un caso de la BC para ser considerado como parte de la solución al nuevo problema. Este valor está definido en dependencia del dominio de los casos donde se maneja siendo su valor dependiente de la información que se almacena, por lo que se define por expertos en el área donde se implementa el sistema.

**Los rasgos.**

Según el tipo de valores del dominio de un rasgo, estos se clasifican en cualitativos o cuantitativos. Los rasgos cualitativos se subdividen en nominales o con orden. En ambos casos los valores pueden representarse por códigos numéricos. Los rasgos con orden se diferencian de los nominales en que entre los códigos se puede establecer una relación de orden. Tanto en los rasgos nominales como los con orden no tiene sentido realizar operaciones aritméticas entre sus valores. Por otra parte los valores dados a rasgos cuantitativos se pueden comparar pues existe una relación de orden y con ellos se puede operar matemáticamente.

### 2.1.1 Técnicas para la adquisición de conocimientos.

Para la definición de los rasgos que conformarán la BC, 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 SBC, en él, se hace necesario la estructuración y el conocimiento del experto humano. (27) Esto trae consigo un gran trabajo por parte del ingeniero de conocimiento al enfrentarse a un área de dominio del experto humano. Por esto se debe seleccionar las técnicas necesarias que permitan obtener el conocimiento de este experto. Entre las técnicas de extracción de conocimiento que se aplican a la creación de las BC se encuentran las siguientes (27):

**Observación directa:** es una técnica de extracción del conocimiento que consiste en la observación simple del modo en que un experto se enfrenta a los problemas del dominio.

**Entrevistas no estructuradas y semiestructuradas:** se basa en realizar una entrevista con los expertos para investigar como representan mentalmente sus conocimientos y como analizan la información inconsistente o imprecisa.

**Conocimiento documentado:** consiste en realizar la extracción de conocimiento de diversas fuentes implicadas (libros, artículos de revistas, manuales, informes y documentos de ayuda), en formato digital o impreso, ya que cualquier documento podría ser útil para la introducción de conocimiento en la base de conocimiento.

La técnica seleccionada para realizar el análisis de las características que influyen en la duración de un proyecto de desarrollo de software fue conocimiento documentado. La fundamentación de la selección se basó en la gran cantidad de documentación que aborda la gestión del tiempo y la duración de proyecto, y en la existencia de estudios previos realizados y publicados en la UCI sobre las estimaciones de tiempo. Reunida toda la información, se revisó para familiarizarse con el contenido y archivarla para su posterior consulta ante cualquier necesitad.

## 2.2 Selección de indicadores.

Para describir el conocimiento se partió de las características que influyen en la duración de un proyecto, teniendo como punto de partida que el tiempo es dependiente del tamaño y del esfuerzo a emplear para alcanzar los objetivos del proyecto. Algunos de los métodos seleccionados fueron a partir de los criterios, según la bibliografía consultada, de ser los más utilizados, mejor referenciados y documentados. Tal es el caso de: Puntos por caso de uso, Puntos de función, COCOMO, PROBE. Además se decidió incluir el método de estimación UCI por tener incidencia directa en el área en la que se enmarca la investigación.

Se utilizó una descripción por categoría, la cual permite obtener con exactitud los indicadores que influyen sobre un proyecto y poder unir los distintos métodos con los indicadores que se repiten o agrupar los que se refieren a un mismo tema. Para obtener una mayor comprensión del dominio de los indicadores y categorías estas se agrupan por el área donde influyen sobre el tiempo, el tamaño y el esfuerzo.

### 2.2.1 Puntos por Casos de Uso.

Para aplicarse, el método Puntos por Casos de Uso (PxCU) depende de casos de usos bien estructurados y bien escritos, con un nivel conveniente de detalle textual. Su objetivo es obtener un número único que caracterice completamente al sistema y que se correlacione con la productividad observada del ingeniero.

La idea central es estimar (cuantificar) el tamaño del software a partir de los requerimientos de los casos de uso. El método cuenta con varios pasos que describen la forma en que este método calcula el tamaño

e incluye un factor cliente que influye en este. Esta medida de tamaño es solo aplicable para las metodologías que trabajan con casos de uso por lo que este factor no es comprendido dentro de la selección de indicadores.

El método PxCU incluye entre sus ecuaciones para calcular la duración de un proyecto varios factores de complejidad técnica y varios factores ambiente. Mediante funciones que incluyen el valor que toma el factor y el peso que tiene este en la duración, se determina el tiempo total estimado para un proyecto.

Los factores técnicos refieren las características del sistema que se desea estimar, y tiene en cuenta la complejidad del sistema que se desea construir. Para determinar el peso de los factores técnicos, se apoya de la siguiente tabla, la cual indica aspectos relacionados con la complejidad de los módulos a desarrollar:

| Factor | Descripción |
|---|---|
| **T1** | Sistema distribuido |
| **T2** | Objetivos de performance o tiempo de respuesta |
| **T3** | Eficiencia del usuario final |
| **T4** | Procesamiento interno complejo |
| **T5** | El código debe ser reutilizable |
| **T6** | Facilidad de instalación |
| **T7** | Facilidad de uso |
| **T8** | Portabilidad |
| **T9** | Facilidad de cambio |
| **T10** | Concurrencia |
| **T11** | Objetivos especiales de seguridad |
| **T12** | Acceso directo a terceras partes |
| **T13** | Facilidades especiales de entrenamiento a usuarios |

Tabla 1: Factores técnicos.

Los valores están dados por descripción según su peso en Irrelevante, Medio y Esencial.

Los factores ambientales influyen sobre el esfuerzo a realizar en el proyecto por el equipo de trabajo, contienen aspectos relacionados con las habilidades y experiencias del equipo. Para determinar el peso de los factores ambientales, se apoya de la siguiente tabla, la cual indica aspectos relacionados con las habilidades y experiencias del grupo o equipo de trabajo:

| Factor | Descripción |
|---|---|
| **E1** | Familiaridad con el modelo del proyecto utilizado |

| | |
|---|---|
| **E2** | Experiencia en la aplicación |
| **E3** | Experiencia en orientación a objetos |
| **E4** | Capacidad del analista líder |
| **E5** | Motivación |
| **E6** | Estabilidad en los requerimientos |
| **E7** | Personal de medio tiempo |
| **E8** | Dificultad en el lenguaje de programación |

Tabla 2: Factores ambientales.

Estos factores tienen un peso significativo en la duración de los proyecto, además se reflejan en cualquier condición en que este proyecto se desarrolle. Teniendo en cuenta esto se proponen como indicadores de duración de proyectos.

**Indicadores seleccionados del método PxCU.**

Cuando se seleccionaron dichos indicadores se realizó una decantación ya que algunos de los mismos coincidían con factores de otros métodos. Los factores de complejidad técnica se decantaron con los factores de ajuste de Puntos de Función (**ver subepígrafe 2.1.2**), con Factor ambiente también se realizó una decantación coincidiendo con los multiplicadores de esfuerzo del método COCOMO (**ver subepígrafe 2.1.3**). Los demás se agruparon por categorías generales de factores de complejidad técnica según afectan el tiempo. La lista de indicadores se muestra en la Tabla 3.

| Categoría | Indicadores |
|---|---|
| **Factores de complejidad técnica tamaño** | Tiempo de respuesta |
| | Eficiencia por el usuario |
| | Portabilidad |
| | Concurrencia |
| | Objetivos especiales de seguridad |
| **Factores de complejidad técnica Esfuerzo** | Familiaridad con el modelo del proyecto usado. |
| | Motivación. |
| | Personal media jornada. |

Tabla 3: Indicadores de PxCU.

### 2.2.2 Puntos de Función.

El método Puntos de Función (PF) se basa en la descomposición de un sistema en componentes más pequeños, de tal forma que estos puedan ser medidos y analizados con mayor facilidad (52). El método

intenta enfocar al software desde el punto de vista del usuario y; la medición, se realiza cuantificando la funcionalidad brindada a dicho usuario, partiendo fundamentalmente de diseños lógicos.

**Identificación de grupos lógicos de datos y funciones transaccionales en las aplicaciones a medir.**

Con respecto a los datos existen dos grupos que se identifican al principio: grupos lógicos de datos y transacciones. Los grupos lógicos de datos, están identificados por los archivos lógicos internos y externos y las transacciones, mediante las funciones transaccionales. Una vez definidos y agrupados estos componentes, se puede mostrar la funcionalidad vista desde una perspectiva del usuario. Esta medida de tamaño es una de las más utilizadas en el mundo, pues es incluida en otros métodos porque se puede obtener una visión general del tamaño.

**Determinar el valor del factor de ajuste.**

El método PF aplica un factor de ajuste que se obtiene mediante una ecuación que tiene en cuenta los grados de influencia de las 14 características que se definen para un sistema. Estas características o factores de ajustes obtienen como valor: sin influencia, incidental, moderado, medio, significativo y esencial, estos se muestran a continuación.

- Mecanismos de recuperación y back-up confiables: Describe el grado en que la carga de transacciones influye en el desarrollo de la aplicación.
- Comunicación de Datos: Describe el grado en el que la aplicación reside directamente en el procesador o se utilizan dispositivos distribuidos que se comunican a través de redes de comunicaciones.
- Funciones de Procesamiento Distribuido: Describe el grado en el que la aplicación transfiere datos entre sus propios componentes.
- Performance: Describe el grado en el que las consideraciones sobre el tiempo de respuesta y el rendimiento exigido al proceso afectan al desarrollo de la aplicación.
- Configuración usada rigurosamente: Describe el grado en que las restricciones en el uso de recursos de máquina influyen en el desarrollo de la aplicación.
- Entrada de datos on-line: Describe el grado en que los datos son introducidos mediante transacciones interactivas.

- Factibilidad Operativa: Describe el grado en que la aplicación se ocupa de aspectos operacionales, como el arranque, copias de seguridad y recuperación.
- Actualización de archivos on-line: Describe el grado en que los ficheros internos son actualizados on-line.
- Interfaces Complejas: Describe el grado en que se consideran los factores humanos y facilidad de uso de la aplicación.
- Procesamiento Interno Complejo: Describe el grado en el que la lógica de proceso influye el desarrollo de la aplicación.
- Reusabilidad: Describe el grado en el que la aplicación y sus componentes de código han sido específicamente diseñados, desarrollados y mantenidos para ser utilizables por otras aplicaciones.
- Fácil Instalación: Describe el grado en que la conversión de anteriores plataformas influencia el desarrollo de la aplicación.
- Soporte de múltiples instalaciones: Describe el grado en que la aplicación ha sido desarrollada para múltiples instalaciones en lugares u organizaciones diferentes.
- Facilidad de cambios y amigabilidad: Describe el grado en el que la aplicación ha sido desarrollada para una fácil modificación de su lógica de proceso o estructura de datos.

**Indicadores seleccionados del método Puntos de Función.**

Puntos de Función permite medir el tamaño del software que va a ser realizado, en este método el factor de ajuste influye considerablemente en el tiempo debido a que tiene en cuenta una serie de indicadores que evalúan la aplicación que está siendo medida y su entorno. La medida de tamaño definida por este se sustituye por la propuesta del método COCOMO debido a que este permite llevarlo a líneas de código (**ver subepígrafe 2.1.3)**.

De esta forma se obtuvieron los indicadores de proyecto que afectan la estimación del tiempo según Puntos de Función. Estos se agregaron en los factores de complejidad técnica de PxCU (**ver epígrafe 2.2.1**) debido a que reflejan la complejidad técnica del sistema. Además se le agregaron los posibles valores con que pueden influir sobre un proyecto. La lista de indicadores se muestra en la Tabla 4.

| Categoría | Indicadores |
|---|---|
| **Factor de Ajustes** | Mecanismos de recuperación y back-up confiables |
| | Comunicación de Datos |

| | |
|---|---|
| | Funciones de Procesamiento Distribuido |
| | Configuración usada rigurosamente |
| | Entrada de datos on-line |
| | Actualización de archivos on-line |
| | Interfaces Complejas |
| | Procesamiento Interno Complejo |
| | Reusabilidad |
| | Facilidad de cambios y amigabilidad |

Tabla 4: Indicadores de Puntos de Función

### 2.2.3 COCOMO.

Este modelo incluye 3 sub-modelos: el modelo de Composición de Aplicaciones, el modelo de Diseño Temprano y el modelo Post-Arquitectura. El modelo de Composición de Aplicaciones es utilizado en proyectos de software que se construyen a partir de componentes pre-empaquetados. Los modelos de Diseño Temprano y Post-Arquitectura se basan en la misma filosofía a la hora de hacer una estimación. Su principal diferencia se produce en la cantidad y detalle de la información que se utiliza para obtener la estimación en cada uno de ellos. El modelo Diseño Temprano se utiliza en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software de un proyecto. En esta fase no se sabe lo suficiente como para dar soporte a la estimación. El modelo Post-Arquitectura es el más detallado y permite mayor precisión en los cálculos de estimación, por lo que fue tomado para el análisis del método y la selección de indicadores.

**Métricas de software.**

En la estimación del tamaño de software COCOMO II utiliza tres técnicas: Puntos de Objeto (PO), Puntos de Función No Ajustados (PFNA) y Líneas de Código Fuente (LOC de "*Line Of Code"*). De las técnicas de obtención del tamaño que propone COCOMO II usar para estimar la duración de un proyecto, los PO son los menos utilizados, estos permiten describir por separado el tamaño de lo que se desea construir con lo que se va a reusar. Los PFNA y las LOC son las técnicas más utilizadas.

Los PFNA son obtenidos de igual forma que el método de estimación PF sin realizar el proceso de ajuste. Como paso adicional COCOMO II convierte los PFNA a LOC considerando el lenguaje de implementación. Esto se realiza para los modelos Diseño Temprano y Post Arquitectura teniendo en

cuenta el **Anexo 1** propuesto por Jones. Siendo las LOC la métrica general para definir el tamaño de un proyecto.

COCOMO II considera como desperdicio al porcentaje de código que se debe eliminar debido a la volatilidad de los requerimientos. Este se usa para ajustar el tamaño efectivo del software a ser desarrollado a los efectos del proceso de estimación. Este desperdicio influye en la ecuación de cálculo del esfuerzo mediante una constante y sirve para ajustar esta función, no teniendo un gran peso en la estimación.

**Modelo de Reuso.**

El modelo COCOMO II permite tener en cuenta si un proyecto de software va a ser construido a partir de componentes existentes. Por lo que permite separar las líneas de código en ESLOC que representa la cantidad equivalente de nuevas líneas de código a desarrollar.

Además incluye otros parámetros que integra para calcular el reuso que va a tener el proyecto:

- Cantidad de líneas de código fuente del software existente usadas para desarrollar el nuevo producto.
- Porcentaje del diseño del software que requiere modificación para alcanzar los objetivos del nuevo software a desarrollar.
- Porcentaje del código del software que requiere modificación para lograr los objetivos del nuevo software a desarrollar.
- Porcentaje del esfuerzo requerido para integrar y testear el software adaptado al producto global.
- Porcentaje de comprensibilidad del software existente. Se determina en función a tres características: estructura, claridad y descriptividad. **Ver Anexo 2**.
- Grado de Evaluación y Asimilación. Porcentaje de esfuerzo necesario para determinar si un módulo de software a adaptar es apropiado a la aplicación, como también para integrar su descripción a la descripción total del producto. **Ver Anexo 3**.
- Nivel de familiaridad del programador con el software. **Ver Anexo 4**.

Estos parámetros influyen directamente sobre los proyectos con reuso. Además que brinda información de el grado con que influye en un proyecto. Teniendo en cuenta que muchos de los proyectos realizados en la actualidad parte de una base reusable, se puedo considerar a estos como indicadores de duración de proyectos de software.

**Reingeniería y Conversión.**

El enfoque de reingeniería y conversión involucra la estimación de un nuevo parámetro que representa el porcentaje de código que sufre un proceso de reingeniería mediante el uso de una herramienta de traducción automática.

El método COCOMO propone factores que influyen exponencialmente en la productividad y esfuerzo de un proyecto de software lo cual tuvo un gran peso para su selección como indicadores.

**Precedencia y Flexibilidad en el Desarrollo.**

El factor de precedencia (PREC) toma en cuenta el grado de experiencia previa en relación al producto a desarrollar, tanto en aspectos organizacionales como en el conocimiento del software y hardware a utilizar, así como la necesidad de innovar.

El factor de flexibilidad (FLEX) considera el nivel de exigencia en el cumplimiento de los requerimientos preestablecidos, plazos de tiempos y especificaciones de interface.

El modelo COCOMO II presenta la tabla de Factores de Escala que están relacionados al modo de desarrollo (**ver Anexo 5**), en la cual se detallan las características que definen los valores de los factores PREC y FLEX.

Estas características muestran en gran medida el grado de complejidad y conocimiento del desarrollo del proyecto. Fueron seleccionadas estas características como indicadores que influyen significativamente en el esfuerzo a realizar en el proyecto.

**Arquitectura y Determinación del Riesgo.**

Este factor involucra aspectos relacionados al conocimiento de los elementos de riesgo crítico y al modo de abordarlos dentro del proyecto. A continuación se muestra un listado de las características que influyen en relación a los riesgos descrita por COCOMO II (19). El **Anexo 6** muestra una tabla de estos factores con los valores que pueden representarla.

- Planificación de la administración de riesgo, identificando todos los ítems de riesgo y estableciendo hitos de control para su solución por medio de la revisión del diseño del producto (PDR de "*Product Design Review*").

- Cronograma, presupuesto e hitos internos especificados en el PDR, compatibles con el plan de administración de riesgo.
- Porcentaje del cronograma dedicado a la definición de la arquitectura de acuerdo a los objetivos generales del producto.
- Porcentaje de arquitecturas de software disponibles para el proyecto.
- Herramientas disponibles para resolver ítems de riesgo, desarrollando y verificando las especulaciones de arquitecturas.
- Nivel de incertidumbre en factores claves de la arquitectura: interfaces de usuario, hardware, tecnología, performance.
- Cantidad y grado de criticidad de ítems de riesgo.

**Cohesión del Equipo.**

El factor de cohesión del equipo tiene en cuenta las dificultades de sincronización entre los participantes del proyecto: usuarios, clientes, desarrolladores, encargados de mantenimiento, entre otros. Estas dificultades pueden surgir por diferencias culturales, dificultad en la conciliación de objetivos, falta de experiencia y familiaridad con el trabajo en equipo (19). La Tabla 5 muestra la descripción de características que influyen en la cohesión de equipo.

| Características | | | | | | |
|---|---|---|---|---|---|---|
| **Compatibilidad entre los objetivos y culturas de los integrantes del equipo** | Poca | Alguna | Básica | Considerable | Fuerte | Total |
| **Habilidad y predisposición para conciliar objetivos** | Poca | Alguna | Básica | Considerable | Fuerte | Total |
| **Experiencia en el trabajo en equipo** | Ninguna | Muy Poca | Poca | Básica | Considerable | Vasto |
| **Visión compartida de objetivos y compromisos compartidos** | Ninguna | Muy Poca | Poca | Básica | Considerable | Amplia |

Tabla 5: Componentes del factor Cohesión del Equipo.

En la UCI, un proyecto puede estar formado por estudiantes, profesores o especialistas vinculados a la producción. Cada uno de ellos puede desempeñar diferentes roles, tales como programadores, analistas, gestores de configuración, diseñador y administrador de bases de datos, líderes de módulos o subsistemas, especialistas en gestión y calidad, entre otros, lo que hace que estas características tengan una gran influencia en la duración de proyectos de esta institución.

**Madurez del Proceso.**

El procedimiento para determinar la madurez del proceso se basa en el Modelo de CMM (Modelo de Madurez de Capacidades o Capability Maturity Model) propuesto por el Software Engineering Institute (SEI) (53). Existen dos formas de calcularlo:

- La primera captura el nivel de madurez de la organización, resultado de la evaluación según CMM. Para aplicar esta medida las empresas deben tener certificado un nivel de CMM por lo que no se tomó en cuenta para la selección de indicadores.
- La segunda está basada en las dieciocho Áreas de Procesos Claves (KPAs) del modelo del SEI. Estos procesos pueden ser descritos por cualquier proyecto independientemente del nivel de CMM que tenga la institución y se describen en la tabla 6.

Estos procesos se clasifican en Casi Siempre (CS), A Menudo (AM), la Mitad de la Veces (MV), Casi Nunca (CN), No se Aplica (NA), No se Conoce (NC).

| Administración de Requerimientos |
|---|
| Planificación del Proyecto de Software |
| Seguimiento y supervisión del Proyecto de Software |
| Administración de Subcontratos |
| Aseguramiento de la Calidad |
| Administración de la Configuración |
| Objetivo del Proceso de Organización |
| Definición del Proceso de Organización |
| Programa de Entrenamiento |
| Administración Integrada de Software |
| Ingeniería del Producto |
| Coordinación entre Grupos |
| Revisión por Pares |
| Administración Cuantitativa |
| Administración de la Calidad |
| Prevención de Defectos |
| Administración de las Tecnologías de Cambio |
| Administración de los Procesos de Cambio |

Tabla 6: Nivel de cumplimiento de los objetivos de cada KPA

**Factores Multiplicadores de Esfuerzo.**

El esfuerzo nominal de desarrollo de un proyecto de software se ajusta para una mejor estimación mediante factores que se clasifican en cuatro áreas: Producto, Plataforma, Personal y Proyecto. El método de COCOMO contempla los aspectos a tener en cuenta para clasificar el factor, valorándolos de Muy bajo, Bajo, Normal, Alto, Muy Alto y Extra Alto (19).

**Factores del producto:** Se refieren a las restricciones y requerimientos sobre el producto a desarrollar.

- Confiabilidad requerida: Este factor mide la confiabilidad del producto de software a ser desarrollado.
- Tamaño de la base de datos: El esfuerzo requerido para desarrollar un producto de software está relacionado con el tamaño de la base de datos asociada. Esto se muestra principalmente en la preparación de los lotes de prueba que se usan en el testeo del producto.
- Complejidad del producto: Analiza la complejidad de las operaciones empleadas en el producto, clasificadas en operaciones: de control, computacionales, dependientes de los dispositivos, de administración de datos y de administración de interfaz de usuario.
- Requerimientos de reusabilidad: Este factor considera el esfuerzo adicional necesario para construir componentes que puedan ser reusados dentro de un mismo proyecto o en futuros desarrollos. Este se refleja en: la creación de diseños genéricos de software, la elaboración de mayor cantidad de documentación, el testeo intensivo para asegurar que los componentes estén debidamente depurados, entre otros.
- Documentación acorde a las diferentes etapas del ciclo de vida: Se evalúa en función de la adecuación de la documentación del proyecto a las necesidades particulares en cada etapa del ciclo de vida.

**Factores de la plataforma:** La plataforma es la infraestructura base de hardware y software que permite desarrollar el producto, aquí se incluye las máquinas virtuales, sistemas operativos, la red o el hardware de la computadora en dependencia del tipo de aplicación que se desea desarrollar. La plataforma incluye cualquier compilador o ensamblador utilizado para desarrollar el sistema, abarcando todas las áreas del desarrollo se detalla a continuación su selección:

- Volatilidad de la plataforma: Este factor se usa para representar la frecuencia de los cambios en la plataforma subyacente.
- Restricción del almacenamiento principal: Este factor representa el grado de restricción del almacenamiento principal impuesto sobre un sistema de software. Se refiere como al almacenamiento

principal al almacenamiento de acceso directo, tales como circuitos integrados, memoria de núcleos magnéticos, excluyendo discos, cintas, entre otros.

- Restricción del tiempo de ejecución: Este factor representa el grado de restricción de tiempo de ejecución impuesta sobre el sistema de software. Es expresado en el tiempo de ejecución disponible que se espera que sea consumido por el sistema o subsistema, del total del recurso formado por el tiempo de ejecución.

**Factores del personal:** Estos factores están referidos al nivel de habilidad que posee el equipo de desarrollo y se aplica a cualquier grupo de personal.

- Capacidad del analista: Se entiende por analista a la persona que trabaja con los requerimientos, en el diseño global y en el diseño detallado. En este análisis no se tiene en cuenta el nivel de experiencia.
- Capacidad del programador: Se mide la productividad por la habilidad del programador en el uso de las herramientas. También se evalúa la capacidad de los programadores para el trabajo en equipo más que para el trabajo individual, resaltando las aptitudes para comunicarse y cooperar mutuamente.
- Continuidad del personal: Este factor mide el grado de permanencia anual del personal afectado a un proyecto de software.
- Experiencia en la aplicación: Este factor mide el nivel de experiencia del equipo de desarrollo en aplicaciones similares.
- Experiencia en la plataforma: COCOMO II reconoce la importancia del conocimiento de nuevas y potentes plataformas, interfaces gráficas, base de datos, redes, etc.
- Experiencia en el lenguaje y las herramientas: Este factor mide el nivel de experiencia del equipo en el uso del lenguaje y herramientas a emplear.

**Factores del proyecto:** Estos factores se refieren a las condiciones y restricciones bajo las cuales se lleva a cabo el proyecto. De estos factores se seleccionó Cronograma requerido para el desarrollo que mide la restricción en los plazos de tiempo impuesta al equipo de trabajo. Otros como multisitio y el uso de herramientas de software se encuentran explícitos en factores de complejidad técnica.

**Selección de los indicadores del método COCOMO.**

Tras el análisis de cada uno de estos elementos se obtuvieron los indicadores de proyecto que afectan la estimación del tiempo según COCOMO. Estos se agruparon por categorías generales. El método sustituye puntos de función por líneas de códigos para medir el tamaño de la aplicación. La lista de indicadores seleccionado, por su extensión, se muestra en el **Anexo 7**.

### 2.2.4 PROBE.

El método PROBE utiliza como unidad de medida LOC. En este método no se toman en cuenta las líneas en blanco o las líneas de comentarios. Este es un método de estimación basado en Proxy que permite utilizar cualquier elemento siempre que cumpla con las condiciones de un proxy. Un proxy es una característica del programa que es fácilmente visualizable en etapas tempranas del desarrollo, visualiza el número de líneas de un programa que apenas se está planeando y su objetivo es estimar las líneas de código de un programa antes de implantarlo. (54)

**Cálculo del proxy.**

PROBE considera dos tipos de partes reutilizadas. En primer lugar están las partes tomadas de una biblioteca de reutilización. Mientras que por otra parte se encuentran las nuevas piezas reutilizables, las cuales son algunas de las partes añadidas que ya han sido estimadas. La categoría de reutilizado es sólo para partes sin modificaciones.

PROBE integra estos factores y calcula la cantidad de LOC estimadas para el nuevo programa incluyendo los términos adicionales base (LOC que han sido añadidas al programa de partida), nuevos objetos (nuevos objetos que se desarrollan), y LOC modificadas se definen teniendo como referencia el formulario de datos presentado.

**Puntos de coincidencia de PROBE con otros métodos de estimación.**

Obtenida la medida de tamaño del método PROBE, se realizó un estudio de los indicadores que tiene en cuenta. Se partió de un análisis de los principales puntos de integración con los otros métodos de estimación, para lo cual se revisaron los trabajos (55) (56) (57) donde se obtuvieron coincidencias en relación a la selección de los indicadores obtenidos de otros métodos. Estos puntos de coincidencia se muestran a continuación.

- Multiplicadores de esfuerzo (**ver subepígrafe 2.2.3**).

- Madurez en el proceso (**ver subepígrafe 2.2.3**).
- Componentes reusables (**ver subepígrafe 2.2.3**).

El método integra todos sus factores a los obtenidos por COCOMO validando la selección de los mismos en este método y no agregándose nuevos indicadores a la selección general.

### 2.2.5 Método de estimación UCI

Las bases del método de estimación de la UCI, se recogen del trabajo publicado por la Ing. Dayami Rodríguez Brito (21) sobre el método de estimación UCI realizado en el 2010. Este cuenta con un grupo de especificaciones para guiar el proceso de estimación en la institución.

El primer paso del método es la identificación de la cantidad de paquetes funcionales que potencialmente tendrá el sistema a desarrollar. Estos paquetes funcionales engloban una serie de funcionalidades y estarán compuestos por una determinada cantidad de Puntos de Función (PF) que será estimada por el método.

Obteniéndose los PF se puede obtener las líneas de código necesarias para medir el tamaño que va a tener el software a desarrollar. Coincidiendo con la selección del indicador de tamaño seleccionado con anterioridad (**ver subepígrafe 2.2.3**)

**Métricas de Tamaño**

Para unificar los puntos de función según la clasificación de los paquetes funcionales, se aplica una métrica donde influye el Factor Cliente (FC). Obtenido el valor del tamaño para cada tipo de módulo (Grande, Mediano, Pequeño) se procede a calcular el tamaño total.

TPF = TotPF *CMax *FC*100

TPF: Tamaño paquetes funcionales.

TotPF: Total de paquetes funcionales.

CMax: Cota Máxima de puntos de función para paquetes funcionales grandes.

Esta función tiene en cuenta los tamaños que pueden tener cada paquete funcional. El tamaño de cada paquete es una medida de apreciación de la persona que está estimando y su valor depende de la

rigurosidad con que esta la estime, siendo un valor que puede variar de una persona a otra lo cual dificulta su precisión en la estimación.

La métrica de tiempo permite obtener como resultado final la estimación de tiempo de un proyecto de desarrollo de software e influyen en este una serie de factores que se describen a continuación:

**Factor del Cliente:** Permite determinar un nivel de incertidumbre asociado a la madurez que tiene el cliente en materia de asimilación de proyectos informáticos, se utilizará para determinar la cantidad de puntos de función que se dejan como reserva. Factores (21):

- Existencia de un sistema anterior: este factor mide si el proyecto que se va a realizar se ha automatizado anteriormente o en otro momento.
- Existencia de resultados de informatización: este factor mide si existen algunas funcionalidades o paquetes funcionales informatizados.
- Existencia de infraestructura tecnológica en la organización: analiza si existe la infraestructura tecnológica que representa la integración de un conjunto de elementos de hardware, software y servicios que en conjunto dan soporte a las aplicaciones de una empresa u organización.
- Existencia de una base legal en la organización: mide si la organización cuenta con una base fundamentada jurídicamente y apoyada en la ley.
- Estructura clara en la organización: este factor se usa para saber si los procesos y áreas funcionales están claramente definidos.
- Disponibilidad de un especialista para atender al proyecto: este factor mide si en la organización se cuenta con un especialista en disposición de atender el proyecto.
- Definidas las funciones de las áreas: analiza que cada una de las actividades que se realizan en cada área están definidas.
- Estabilidad de los requisitos: mide si los requisitos funcionales y no funcionales se mantendrán estables y sin posibilidad de cambios.
- El cliente es el usuario de la aplicación: este factor se usa para saber si el cliente que solicita la realización del proyecto es el usuario que interactuará con la aplicación.

**Factor Ambiente:** Factor de Ambiente (FA): es interno, se determina a través de analizar las condiciones de la facultad en la que se desarrolla el proyecto, no influye en el precio final, pero si en las valoraciones

para la realización del proyecto y se utilizará como indicador para saber si la facultad ha creado condiciones para cumplir los compromisos. Se aplicará en el precio de negociación y previo a la contratación. Factores (21):

- Temática conocida por el centro: este factor mide si la estructura de producción del centro tiene experiencia en el desarrollo de proyectos sobre esta temática o cuenta con personal capacitado en esta temática.
- Conocimientos del centro sobre Base de Datos, lenguaje de programación, control de versiones: mide si la estructura de producción del centro domina los conocimientos de Base de Datos, los conocimientos sobre el lenguaje de programación que especifica el cliente o con el que se pretende realizar el proyecto y los conocimientos y herramientas para el control de versiones que es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.
- La facultad tiene conocimientos sobre la Gestión de Proyectos: la Gestión de Proyectos se encarga de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y coste definidos. Este factor mide si la estructura de producción tiene experiencia en el desarrollo de proyectos exitosos.
- Proyecto de continuidad: este factor analiza si se ha desarrollado una parte del proyecto y se pretende continuar el desarrollo.
- Disponibilidad del centro de líder de Proyecto: mide si la estructura de producción del centro cuenta con personal con experiencia en la gestión de proyectos.

**Factor de Complejidad Técnica:** Es asociado a la complejidad de la tecnología y requerimientos no funcionales del sistema, influye directamente en el esfuerzo y fundamentalmente en las etapas de implementación. Factores (21):

- Rendimiento de la Aplicación: este factor tiene en cuenta la velocidad de procesamiento, el tiempo de respuesta, consumo de recursos, rendimiento efectivo total y eficacia de la aplicación.
- Rendimiento de la Plataforma: este factor tiene en cuenta la velocidad de procesamiento, el tiempo de respuesta, consumo de recursos, rendimiento efectivo total y eficacia de la plataforma.
- Complejidad de Diseño: mide el nivel de complejidad en el conjunto de operaciones técnicas-proyectuales necesarias para la información visual, al objeto de dotarla de la mayor cantidad posible

de atributos eficaces, comprensibles y persuasivos para la fácil y completa percepción del mensaje a transmitir.

- Seguridad: este factor tiene en cuenta la disponibilidad de mecanismos que controlen o protejan los programas o los datos.
- Integración.
- Asimilación de Sistemas.
- Asimilación de Estándares.

**Selección de los indicadores del método de estimación UCI**

El tamaño puede ser expresado en PF y este en LOC, por lo que se mantiene la selección realizada con anterioridad. Se realizó un análisis de los factores clientes y su influencia sobre la duración, se determinó la selección de todos los factores exceptuando el factor el cliente es usuario de la aplicación y existencia de resultado de informatización debido al poco peso que ejerce sobre la duración.

Los factores ambientes que presenta este método, contiene una descripción detallada de los procesos productivos de la UCI, pero estos contienen valoraciones que no influyen en la duración final, son utilizados para saber si la facultad ha creado condiciones para cumplir con los compromisos.

No se tuvo en cuenta en los factores de complejidad técnica indicadores que coinciden con la selección previamente hecha por los autores. Estos indicadores son procesamiento interno complejo, reutilización y asimilación de dispositivos. La selección se muestra en el **Anexo 8**.

### 2.2.6 Indicadores de Recursos.

Terminado el análisis de los principales métodos se realizó un estudio de otros factores que se pueden tener en cuenta para la selección de indicadores. Se realizó un estudio basado en el trabajo de tesis de grado de Yadira Machado y Yairelis Peña (3) en el que se propusieron una serie de indicadores relacionados con la tecnología, los cuales influyen en el esfuerzo realizado por el equipo del proyecto. Este cuenta con una serie de indicadores que aportan los recursos utilizados en el proyecto, los cuales se tienen en cuenta también para realizar la estimación. Estos indicadores se agruparon en la categoría reingeniería y conversión.

Indicadores de Recursos

- Metodología de desarrollo de software.
- Estilo Arquitectónico.
- Patrón de Arquitectura.
- Herramienta de Modelado Visual.
- Herramienta de Programación.
- Herramienta Sistema Gestor Base de Datos.
- Herramientas utilizadas en la Planificación.
- Herramientas utilizadas en la Gestión de Configuración.
- Herramientas utilizadas en la Estimación.
- Cantidad de PC.

Se incluyeron indicadores que influyen en el esfuerzo como son: cantidad de especialistas, cantidad de estudiantes y tipo de proyectos. Estos indicadores se ajustan a características de la UCI debido a que el tiempo que dedica un estudiante a realizar una tarea no es el mismo que la de un especialista, además, se cuenta con una gran diversidad de proyectos con características distintas. A la categoría Cohesión de Equipo se le agregó el indicador comunicación entre los miembros del equipo como medidor del resultado de la cohesión.

Se agregó nacionalidad del cliente a los indicadores de factor cliente. Se considera que el mismo es un indicador que puede influir en la duración de un proyecto debido a la lejanía, afectando el proceso de comunicación, levantamientos de requisitos, despliegue del proyecto, entre otros.

**2.3 Indicadores propuestos.**

Los indicadores propuestos partieron de una selección de los indicadores de los métodos anteriormente explicados y bibliografía revisada. Primeramente se buscó los que eran semejantes, y se solaparon. Se buscaron puntos de coincidencia entre ellos, para esto se revisaron también trabajos relacionados con la integración de las teorías de Boehm y Humphrey (55) (56) (57), los cuales hacen referencia a los puntos de integración entre el método PROBE y COCOMO.

La selección se realizó con el objetivo de obtener 3 grupos diferentes de indicadores. El primero son todos los indicadores que influyen sobre un proyecto de desarrollo de software. Debido a la existencia de indicadores con características comunes y que afectan una determinada parte del proyecto, se realizó una

selección por categorías. En cada categoría se agrupa un grupo de indicadores de proyecto. Por último se agruparon estas categorías en dependencia de la variable en la que más incidencia tiene según la afectación del tiempo de desarrollo, para realizar la estimación. El **Anexo 9** y **Anexo 10** muestran la selección íntegra de los indicadores de tamaño y esfuerzo respectivamente que influyen en la duración de un proyecto.

### 2.4 Validación por expertos.

La validación de los indicadores propuestos se realizó a través de criterios de expertos, con la que se pretende tener estimaciones de precisión razonable sobre la validez de la selección. Para que una persona sea considerada experta debe, poseer un conocimiento profundo de la tarea o actividad que será analizada y estar familiarizado con la misma. (58) Para realizar la validación se dispuso de una serie de pasos definidos en el trabajo del Dr. Raúl Fernández Aedo (59), este plantea comenzar por la selección de los expertos y medir su coeficiente de competencia, luego elaborar los cuestionarios, elegir el método a utilizar para la obtención de los resultados, ejecutar el método y procesar la información.

Se realizó la selección de los expertos cumpliendo como condición que la persona tuviera conocimientos sobre la gestión de proyectos y como parte de ella la planificación. Se seleccionaron 11 expertos entre los cuales se encuentran, líderes de proyecto, planificadores, miembros del grupo de estimación de Calisoft y especialistas de calidad y gestión de software, todos con más de 2 años de experiencia. A estos se le aplicó una encuesta (**ver Anexo 11**) de manera individual para determinar el coeficiente de competencias y medir el nivel de conocimiento que posee esa persona para ser considerado como experto en esta investigación.

#### Cálculo del coeficiente de competencias.

Después de realizada la encuesta, se procedió a determinar el coeficiente de competencias (K) según define el método mediante la fórmula: $K = (Kc + Ka)/2$ siendo Kc el coeficiente de conocimientos y Ka el coeficiente de argumentación. (59)

Para valorar el nivel de experiencia que poseen se utilizó la primera pregunta del cuestionario. A partir de ahí se calcula el Kc para cada uno de los expertos sobre la base de la valoración del propio experto sobre su conocimiento en una escala del 1 al 10 y multiplicado por 0.1, según define el método. Luego, el Kc se calcula promediando los valores de cada pregunta.

Para valorar los aspectos que influyen en el nivel de argumentación se utiliza la pregunta 3 del cuestionario y se contrastan las respuestas de los expertos con los valores de la tabla patrón (Ver tabla 7). Esta está definida por el método que refleja el grado de influencia de los conocimientos de los expertos sobre el tema, calculando el Ka a partir de la sumatoria de estos valores.

| No. | Fuentes de argumentación | Grado de influencia | | |
|---|---|---|---|---|
| | | Alto | Medio | Bajo |
| **1** | Análisis realizado por usted | 0.3 | 0.2 | 0.1 |
| **2** | Experiencia | 0.5 | 0.4 | 0.2 |
| **3** | Trabajos de autores nacionales | 0.05 | 0.05 | 0.05 |
| **4** | Trabajos de autores extranjeros | 0.05 | 0.05 | 0.05 |
| **5** | Su propio conocimiento del tema | 0.05 | 0.05 | 0.05 |
| **6** | Su intuición | 0.05 | 0.05 | 0.05 |
| **7** | Total | 1 | 0.8 | 0.5 |

Tabla 7: Tabla patrón.

Posteriormente obtenidos los resultados se valoran de la manera siguiente:

- Si 0.8 < K < 1 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

Solamente se tuvo en cuenta para la consulta, los expertos que posean competencia alta o media. El resultado del coeficiente de competencias del cuestionario aplicado puede verse en el **Anexo 12**. Se muestra a continuación (**ver figura 2**) el resultado general del coeficiente de los 11 expertos que colaboraron, de los cuales 3 tienen categoría científica de máster con más de 5 años de experiencia representando el 27%, 3 son líderes de proyecto con más de 4 años de experiencia representando también el 27% y 5 asesores o especialistas de calidad y del departamento de pruebas de software representando el 45%.

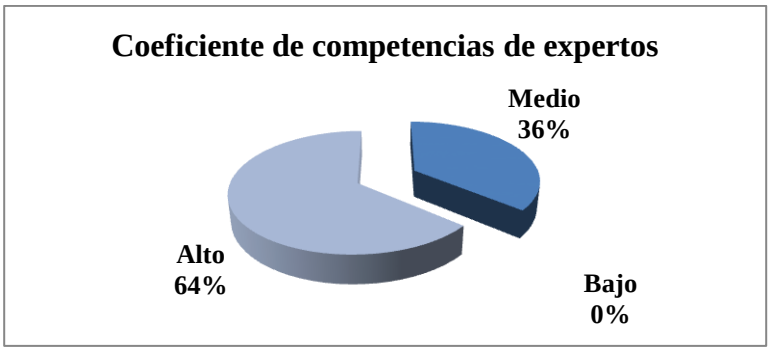


Figura 2: Coeficiente de competencia de expertos.

Luego se realizó la elaboración y lanzamiento de la encuesta de selección de indicadores (**ver Anexo 13**). La encuesta se elaboró de manera que facilitara las respuestas por parte de los consultados. En ella se evalúa el grado de factibilidad de cada indicador por separado para contribuir a la estimación de manera general, además el grado de factibilidad de cada indicador tanto para la categoría a la que pertenece y finalmente el grado de factibilidad de cada indicador para la clasificación de tamaño y esfuerzo. La valoración se hace en una escala del 1 al 10 donde el 10 es el valor máximo de factibilidad.

**Desarrollo práctico y explotación de resultados.**

Para el análisis estadístico de los resultados se utilizó el método T-Student. Este permite realizar un análisis estadístico basado en el cálculo de estadísticos descriptivos previos: el número de observaciones, la media y la desviación típica en cada grupo. Se utilizó debido a que su función es comparar 2 grupos de puntuaciones (medias aritméticas) y determinar las diferencias que existan entre las muestras, permitiendo una rápida forma de comprensión de los datos estadísticos.

El procesamiento de los datos de los cálculos se hizo mediante la herramienta SPSS versión 13.0 (Statistical Product and Service Solutions). ¨Esta herramienta proporciona un poderoso sistema de análisis estadístico y de gestión de datos en un entorno gráfico, utilizando menús descriptivos y cuadros de diálogos.¨ (60) Se consideraron como valores de entrada, los datos obtenidos como resultado de las encuestas de los especialistas.

Se definió el 7 como valor para la prueba debido a que los valores en los que oscilan las respuestas de los expertos están entre 1 y 10 siendo 10 el valor máximo. Se considera Adecuado cuando se obtiene un

valor superior a este nivel y si el valor se encuentra por debajo No adecuado, considerando las siguientes hipótesis:

**$H_0$: μ = 7** Adecuado

**$H_1$: μ ≠ 7** No adecuado

**Resultado final de la validación de indicadores propuestos.**

La interpretación de los resultados se basa en que:

Siendo la Probabilidad asociada (P) definida como Sig. (2-tailed) en el **Anexo 14** y el Nivel de significación (α) donde α = 0.05 definido por el método. Si el resultado de la probabilidad de la prueba P ≥ α o el intervalo de confianza en la tabla contiene los valores de las medias inferior (Lower) y superior (Upper), si contiene al 0 no se rechaza $H_0$ por lo que es Adecuado y si P < α o el intervalo de confianza no contiene al 0 no se rechaza $H_1$ y se evidencian 2 criterios.

Si en el intervalo de confianza los valores tienden al signo negativo existe una desviación a la izquierda y no se rechaza $H_1$ por lo que la hipótesis planteada tiende a ser No adecuado.

Si en el intervalo de confianza los valores tienden al signo positivo existe una desviación a la derecha y no se rechaza $H_0$ por lo que la hipótesis planteada tiende a ser Adecuado.

Al realizar el análisis estadístico de los datos que arrojó la tabla (**ver Anexo 14**) los resultados de la aplicación del método para una media de $H_0$: μ = 7 para un nivel de confianza del 99%, este valor que aparece en la tabla se introduce cuando se va a realizar la prueba y es el porciento de confianza con que se quiere realizar la prueba. Las probabilidades asociadas son mayores que α = 0.05 y el intervalo de confianza contiene al 0, en el caso de los pocos que no lo contenían los valores apuntaban al signo positivo por lo que en ambos casos no se rechaza $H_0$ y se infiere que el grado de factibilidad de cada indicador individualmente para contribuir a la estimación es Adecuado.

El análisis estadístico para definir el grado de factibilidad con respecto a su categoría correspondiente (**ver Anexo 15**) quedó validado como lo anteriormente planteado. Además permitió desglosar mejor las características de los proyectos por categoría y definir qué elementos tienen mayor impacto en la categoría (resaltado más oscuro), además de qué categorías influyen más debido al peso que puedan tener estos indicadores dentro de la misma como se muestra en el **Anexo 16**.

El análisis estadístico para definir el grado de factibilidad para tamaño y esfuerzo (**ver Anexo 17**) quedó validado con lo anteriormente planteado. Se realizó con respecto a la característica que representa un elemento distintivo al tamaño o al esfuerzo ya que de estos aspectos depende la estimación del tiempo, quedando registrado en el **Anexo 18** los principales indicadores de tamaño y esfuerzo.

Se constató la correcta distribución de los indicadores por las categorías y tamaño o esfuerzo. Además se obtuvo el grado de factibilidad que posee cada indicador para la duración de un proyecto de software, mediante el cálculo de la media. El cálculo fue realizado por la herramienta SPSS y se muestran sus resultados en el **Anexo 19.**

## 2.5 Representación del conocimiento.

Obtenidos los indicadores que representan las características de los proyectos, se pasó a la representación del conocimiento, dando paso al diseño de la estructura que conformará la base de casos y de cada uno de sus componentes.

La definición del conocimiento mediante casos comienza por la estructura que estos deben tener. Para la presente investigación los casos están formados por un grupo de rasgos predictores y un rasgo objetivo.

### Rasgos predictores.

El conjunto de rasgos predictores representa el conjunto de indicadores definidos que caracterizan un proyecto. Los rasgos predictores además poseen un dominio de valores, una relevancia y una incertidumbre.

### Dominio de valores

Los valores del dominio están dados por las distintas formas en que se puede presentar un indicador en un proyecto. Estos fueron obtenidos de la selección de indicadores, mediante el estudio de los métodos de estimación y reflejan la manera en que los expertos pueden apreciar su ocurrencia en un proyecto. Los rasgos se agruparon según los valores del dominio en nominales, cuantitativos y ordinales, detallándose los valores que pueden tomar estos últimos, esta información puede ser consultada en el **Anexo 9** y **Anexo 10**.

### Relevancia

La relevancia se definió por la factibilidad de de los indicadores expresado en el **Anexo 19**, definido por valores de 1 a 10 donde los rasgos que más se aproximen a 10 tienen un mayor impacto sobre la duración de un proyecto y más cerca de 1 un impacto menor.

**Rasgo objetivo**

El valor que representa el rasgo objetivo es la duración de los proyectos, dados en meses. Este valor se obtiene de conjunto con el caso que se almacena en la BC representando la solución o el resultado que se obtiene de un proyecto definido por sus rasgos predictores.

**La incertidumbre.**

Para definir los valores de la incertidumbre se tiene en cuenta como esta influye en el dominio. La incertidumbre está definida por valores en el intervalo [0; 1], indicando los valores más cercanos a 1 un mayor nivel de certeza y más cercanos a 0 mayor nivel de incertidumbre. Se toma el intervalo positivo debido a que la estimación se realiza con valores conocidos de ocurrencia.

La incertidumbre de cada rasgo denota el grado de certeza de que este se comporte de cierta manera en un caso específico, por lo que para cada caso sus rasgos predictores tendrán su propia incertidumbre y esta será facilitada por los expertos junto a la entrada de los rasgos que componen un caso de la BC. Para la entrada de los casos en la que el experto no especifique el valor de la incertidumbre de un rasgo se asume que se conoce y se está seguro de su ocurrencia, siendo 1 valor de la incertidumbre.

Al introducir un patrón de búsqueda que no defina un valor de incertidumbre para un rasgo, se puede representar mediante la media ocurrencia de la incertidumbre en los casos almacenados en la BC. Con el cálculo de la media aritmética se obtiene un valor promedio de ocurrencia de la incertidumbre de un rasgo, con esto se provee de este valor al patrón de búsqueda siendo necesario para el proceso de recuperación de los casos semejantes.

La media aritmética de un rasgo ($\bar{x}_j$) se obtiene mediante la siguiente función (61):

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

Donde:

$x_{ij}$: Valor de la incertidumbre del rasgo j en el caso i.

*n:* Cantidad de casos.

**La entrada de los casos.**

Los casos partirán de experiencias que se puedan obtener de proyectos desarrollados en la UCI. En este sentido los casos que almacena el sistema vienen dados por expertos de esta área según ocurran en el tiempo, o casos previamente seleccionados y conformados que representen un hecho consumado. En ambos casos la BC permite la inclusión de los datos teniendo en cuenta los aspectos antes mencionados.

### 2.5.1 Funciones de semejanza

La comparación entre los rasgos está dada por la relación entre los valores de dominio de un rasgo del patrón de búsqueda $O_0$ ($V_1$) y el caso almacenado en la BC $O_t$ ($V_2$). La función de comparación entre rasgos da el resultado en valores en el intervalo de [0; 1], siendo 1 el máximo valor de similitud entre los rasgos y 0 el menor. A continuación se detallan las funciones en dependencia del dominio de los rasgos.

Para los rasgos de dominio nominales se toma la función propuesta en el trabajo (62) del MsC. Sergio M. Rivera, definiendo que la semejanza es 1 si los elementos son iguales, 0 sin no lo son, la función queda de la siguiente forma:

$$\delta_i(O_0, O_t) = \begin{Bmatrix} 1 \rightarrow V_1 = V_2 \\ 0 \rightarrow V_1 \neq V_2 \end{Bmatrix}$$

Para los valores cuantitativos se utiliza la distancia absoluta de Manhattan ajustada al rango, siendo este la diferencia entre el mayor y el menor número conjunto de valores existentes. Se utilizó el ajuste propuesto en el trabajo de Rivera Rodríguez (62) que se muestra a continuación:

$$\delta_i(O_0, O_t) = 1 - \frac{|V_1 - V_2|}{r_{max} - r_{min}}$$

El trabajo de Jiawei Han y Micheline Kamber (63) describe cómo manejar las variables ordinales. El cálculo de similitud incluye los siguientes pasos (63):

1. Cada rasgo *f* posee *M* valores ordinales, los cuales se representan desde 1 hasta $M_f$, remplazando cada valor por su correspondiente en el rango $r_{if} \in [1; .. ; M_f]$.

2. Para poder aplicar la función a los distintos estados que pueden tener las variables ordinales, se debe mantener el rango de cada variable entre los intervalos de 0 a 1 de modo que cada variable tenga el mismo peso:

   $$V_{if} = \frac{r_{if} - 1}{M_f - 1}$$

3. La similitud se calcula con respecto a la distancia entre dos objetos dentro de un conjunto. La distancia Manhattan permite este cálculo y se muestra su función a continuación:

   $$\delta_i(O_0, O_t) = 1 - \left|\left(V_{1f} - V_{2f}\right)\right|$$

La función de semejanza relaciona los valores de cada rasgo para obtener las semejanzas de los casos. En el trabajo de Dra. Martínez, Dra. García, Dra. García (29) plantea el uso de un modelo matemático que calcula la semejanza mediante la sumatoria de la relevancia de cada rasgos con la función de comparación entre rasgos $O_t$ y $O_0$ afectada por la incertidumbre, donde esta incertidumbre se calcula mediante el valor absoluto de la diferencia de las incertidumbre del rasgo almacenado y el patrón de búsqueda restada a la cuota máxima, garantizando la igualdad entre estas. El modelo matemático se muestra a continuación:

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

Donde:

n: Número de rasgos predictores.

$p_i$: Peso o relevancia del rasgo i.

$\mu_i$: 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.

### 2.5.2 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. Por esto el umbral usado en la semejanza de los casos se define por expertos.

Además, se provee el uso de otra función para que pueda ser utilizada en el proceso de recuperación de los casos. Esta función utilizada por la Dra. Natalia Martínez en su trabajo (29) tiene en cuenta los casos de la BC y propone el cálculo del umbral de semejanza basada en la semejanza de los casos de la BC:

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

*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 $O_t$.

**Conclusiones.**

Se definieron los conceptos principales que se abordan en la construcción de las BC. Se realizó un estudio de los métodos de estimación más utilizados que permitió realizar una selección de indicadores que influyen en la duración de un proyecto de desarrollo de software. Se definió una propuesta de indicadores a ser validadas por expertos en el área de planificación y gestión de proyectos. Se conformó, aplicó y analizó una encuesta para medir la competencia de los expertos y otra para validar los indicadores. Se conformaron los rasgos de los casos con los indicadores validados y se definió una relevancia para cada uno de estos. Se definió la estructura de la memoria de la BC así como las funciones de comparación de los rasgos atendiendo a la cantidad de valores de dominio que pudieran tomar y la función de semejanza entre casos. Además se provee de funciones para ser utilizadas en el proceso de recuperación para obtener la incertidumbre y el umbral.

## CAPÍTULO 3: CARACTERÍSTICA Y ANÁLISIS DEL SISTEMA

### Introducción al capítulo.

En este capítulo se describe la propuesta del sistema, los requerimientos funcionales y las listas de reservas del producto. Se describen las historias de usuarios y los planes de iteraciones y de entrega pertenecientes a la fase de exploración y de planificación respectivamente.

### 3.1 Propuesta del sistema.

La Base de Casos para la Representar el Conocimiento Adquirido Relacionado con la Estimación de la Duración de los Proyectos de Desarrollo de Software, almacena la experiencia obtenida de los procesos productivos de la Universidad de las Ciencias Informáticas para que pueda ser utilizada por sistemas basados en conocimiento en la estimación de la duración de proyectos de desarrollo de software. La BC contiene las descripciones de los proyectos en forma de casos, además las características que van a representar la categoría asociada al rasgo y el área (tamaño o esfuerzo) que influye, así como la relevancia de cada uno de estos. Los casos están compuestos por un grupo de rasgos predictores y un rasgo objetivo. Los casos serán entrados por los usuarios que tengan esta responsabilidad. Además el sistema permitirá la administración de los usuarios y sus permisos para que puedan acceder a este y realizar las funciones que le son afines.

Para una mejor comprensión del funcionamiento del sistema se muestra en la figura 3 el modelo de dominio del sistema.

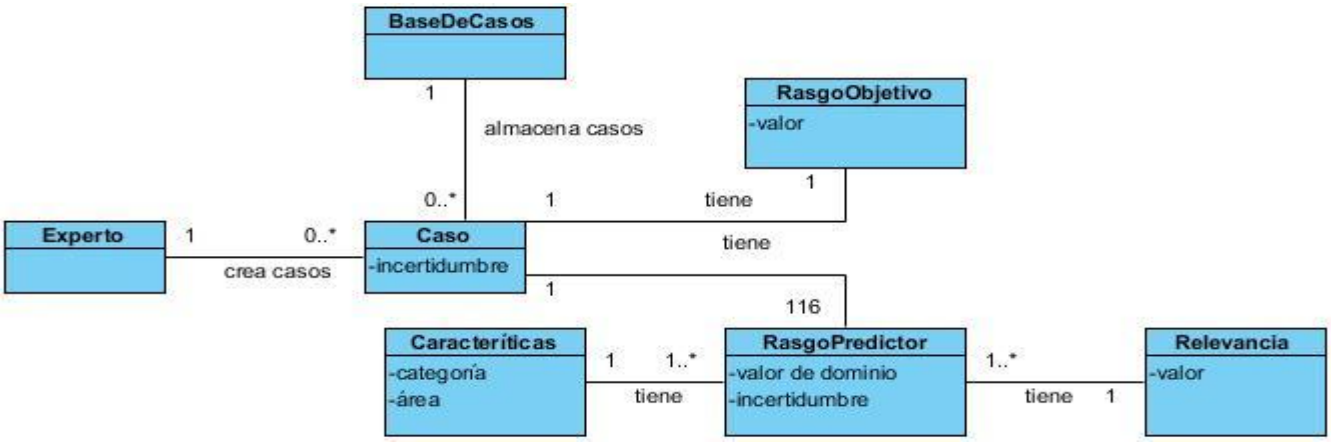

Figura 3: Modelo de dominio del sistema.

### 3.2 Personas relacionadas con el sistema.

Las personas relacionadas con el sistema son aquellas que interactúan con el mismo y obtienen un resultado.

| Personas | Justificación |
|---|---|
| **Experto** | Es un actor del sistema encargado de registrar el umbral y de controlar la información de la categoría y relevancia de los rasgos. |
| **Administrador** | Es un actor del sistema especializado, encargado de controlar la gestión de los usuarios del sistema y los niveles de acceso mediante roles. |
| **Asistente** | Es un actor del sistema que se encarga de realizar el registro de los casos de un proyecto. |

Tabla 8: Personas relacionadas con el sistema.

### 3.3 Fase de exploración.

La metodología de desarrollo XP se inicia con la fase de exploración, en esta fase se obtiene suficiente material de trabajo para producir las historias de usuario. Mientras el equipo de desarrollo se familiariza y prueba la tecnología y herramientas que han sido seleccionadas. En esta fase se estima el tiempo de desarrollo que puede tomar de pocas a varias semanas en dependencia del tamaño y la familiaridad que se tenga con la tecnología. (64)

#### 3.3.1 Historias de Usuario.

Las Historias de Usuario (HU) son la forma en que se especifican los requisitos funcionales de software en la metodología de desarrollo XP, se realiza una por cada característica que deba cumplir el sistema. Las HU constan de varias líneas escritas por el cliente en lenguaje no técnico y el tiempo de desarrollo de la misma.

A continuación se muestran las HU Gestionar caso, las demás se muestran en el **Anexo 20.**

| **Historia de Usuario** | |
|---|---|
| **No: 5** | **Nombre:** Gestionar caso. |
| **Usuario:** Asistente | |
| **Prioridad en el Negocio:** Alta | **Nivel de Complejidad:** Alta |

| | |
|---|---|
| **Estimación: 3 semana** | **Iteración Asignada:** 2 |
| **Descripción:** Permite crear, modificar, listar y ver un proyecto en el sistema. | |
| **Observaciones:** El asistente debe estar autenticado. | |

Tabla 9: HU Gestionar Caso

### 3.3.2 Requisitos no funcionales del sistema.

Los requisitos no funcionales del sistema, son, propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido y confiable.

**Requisitos de Usabilidad.**

- El sistema puede ser usado por personas con conocimientos en el área de gestión de software y gestión del tiempo de proyectos aunque no sean expertos en computación.

**Requisitos de apariencia o interfaz externa:**

- El sistema brindará una interfaz amigable para sus usuarios, fácil de usar, el color de fondo será blanco, la letra tendrá fuente Verdana, tamaño de fuente 11px y alineación Justificada.
- Los errores estarán enmarcados en rojo para que sean visibles al usuario e indicarán las posibles causas.
- Los mensajes estarán enmarcados en azul e indicarán al usuario que se ha realizado una acción sobre el sistema.

**Requisitos de software.**

- La PC 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).
- La aplicación debe tener alojado el sistema gestor de bases de datos PostgreSQL 8.4 o contar con dicho servidor externo. En caso de ser externo el servidor de base de datos, debe tener instalado la misma versión.
- La computadora donde esté alojada la aplicación, debe tener instalado el Apache Tomcat como servidor Web, preferentemente una versión superior a 6.X y la Máquina Virtual de Java v1.7.0.

**Requisitos de Hardware.**

- Es necesario contar con una computadora con un microprocesador con velocidad superior a los 1.6

GHz, y con una memoria RAM superior a 1 Giga Bytes, si en ella no se encuentra el servidor de base de datos. Si se encuentra el servidor en la misma computadora esta debe poseer memoria RAM superior a los 2 Giga Bytes.

- Si el servidor de base de datos es externo, debe estar alojado en una computadora con 1 GB de RAM, más de 150 HDD y tener la red disponible con una velocidad de 10/100/1000 Mb/s.

**Requisitos de Seguridad.**

- El usuario debe autenticarse antes de entrar al sistema, su autentificación será negociada con el servicio, Lightweight Directory Access Protocol (LDAP) de la UCI.
- El sistema debe estar disponible en todo momento para aquellas personas que necesiten acceder a la información, siempre que tengan autorización para hacerlo.

**Requisitos de Portabilidad:**

- Se garantiza que el sistema corre en ambientes multiplataforma a través de la máquina virtual de java.

## 3.4 Fase de planificación.

En la metodología de desarrollo ágil XP la fase de planificación se encarga de planificar el proceso de desarrollo de una aplicación de software, se realiza una estimación del esfuerzo que se empleará para implementar cada HU, determinando un cronograma con la fecha de entrega de las mismas y organizadas por iteraciones.

### 3.4.1 Plan de iteraciones.

Para el desarrollo del plan de iteraciones se seleccionan las historias de usuario que serán desarrolladas en cada entrega de acuerdo al orden establecido. Las iteraciones son cortas y no exceden las 4 semanas en su desarrollo, quedando conformado el plan de iteraciones de la siguiente manera:

| Iteraciones | Orden de las Historias de usuario a implementar | Cantidad de tiempo de trabajo |
|---|---|---|
| **1** | Autenticar usuario.<br>Gestionar usuario.<br>Gestionar rol<br>Gestionar usuario-rol | 4semanas |
| **2** | Gestionar caso.<br>Registrar umbral. | 4 semanas |
| **3** | Gestionar relevancia de los rasgos.<br>Gestionar categoría de los rasgos | 2 semanas |

Tabla 10: Plan de Iteraciones

De esta forma quedaron desglosadas las HU por las iteraciones a realizar por parte del equipo de desarrollo, las cuales tendrá una duración total de 10 semanas.

### 3.5 Plan de Entregas.

El plan de entregas que se presenta a continuación constituye un estimado para la fase de implementación. Como salida se realizarán versiones entregables del sistema al finalizar cada iteración, estas se muestran en la tabla 14.

| Sistema | Final de la Iteración 1 (3ra semana de abril del 2012) | Final de la Iteración 2 (3ra semana de mayo del 2012) | Final de la Iteración 3 (1ra semana de junio del 2012) |
|---|---|---|---|
| Base de Casos para la Representar el Conocimiento Adquirido Relacionado con la Estimación de la Duración de los Proyectos de Desarrollo de Software | Versión 0.1 | Versión 0.2 | Versión 1.0 |

Tabla 11: Plan de Entregas

### Conclusiones.

En este capítulo se describió la propuesta de solución del sistema, se identificaron los requerimientos funcionales y no funcionales, se detallaron las HU que describen las funcionalidades y se planificó el ciclo de iteraciones y el plan de entrega.

## CAPÍTULO 4: DISEÑO, IMPLEMENTACIÓN Y PRUEBA

### Introducción.

En este capítulo se describen los patrones arquitectónicos y de diseño utilizados en la construcción del sistema. Se describen las tarjetas clase-responsables-colaborador, además del modelo físico de la base de datos. Se realiza un desglose de las tareas de ingeniería por cada una de las iteraciones que se definieron para realizar el desarrollo. Se realiza además la descripción de las pruebas al sistema y el resultado de las mismas.

### 4.1 Patrones arquitectónicos.

Los patrones arquitectónicos son plantillas que describen los principios estructurales globales que construyen las distintas arquitecturas de software viables. Plantean una organización estructural fundamental para un sistema de software, expresando un conjunto de subsistemas predefinidos, especificando responsabilidades y organizando las relaciones entre ellos. (65)

#### 4.1.1 Modelo Vista Controlador.

El patrón arquitectónico Modelo-Vista-Controlador (MVC) se ha adoptado por la comunidad de diseño distribuido para aplicarlo en los niveles de la arquitectura a gran escala. El Modelo es la Capa del Dominio, la Vista es la Capa de Presentación, y el Controlador son los objetos del flujo de trabajo en la capa de Aplicación. (66) Grails sigue el patrón MVC, el mismo establece que los componentes del sistema deben organizarse en 3 capas según su misión:

**Modelo**.

Contiene los componentes que representan y gestionan los datos manejados por la aplicación. Generalmente, los objetos encargados de leer y escribir en la base de datos. (67) Se compone por los objetos de dominio de grails, encargados de contener la lógica del negocio de la aplicación; y los servicios, responsables de manejar la lógica de negocio de la aplicación.

Grails define por convención que las clases de dominio se encuentran en la carpeta grails-app/domain y los servicios en la carpeta grails-app/services del directorio raíz de la aplicación, además estos se denotan por su nombre seguido de la palabra “Service”.

**Vista**.

Los componentes de esta capa son responsables de mostrar al usuario el estado actual del modelo de datos, y presentarle las distintas acciones disponibles. (67) En grails, esta capa o componente está definido por las páginas servidoras de grails (gsp por sus siglas en inglés Groovy Service Page) y el conjunto de etiquetas lógicas de presentación (tags), capaces de manejar y mostrar los elementos del negocio necesarios en la capa de presentación.

Grails define que se encuentran en la carpeta Grails-app/views de la raíz del proyecto, las mismas son agrupadas en carpetas que responden a un controlador específico. De esta forma se ahorra tiempo al no especificar el controlador que recibe la información.

**Controlador.**

Contendrá los componentes que reciben las órdenes del usuario, gestionan la aplicación de la lógica de negocio sobre el modelo de datos, y determinan que vista debe mostrarse a continuación. (67) Se compone por los controladores, los cuales especifican que servicios llamar para manejar la lógica del negocio y el envío de las respuestas a las vistas.

Estas clases se encuentran en la carpeta Grails-app/controllers y su denominación está compuesta por el nombre del controlador seguido por la palabra “Controller”.

Para una mejor comprensión de la relación entre cada una de las capas con la arquitectura MVC a continuación se muestra un diagrama de clases del diseño correspondiente a la tarea No. 3 Ver Usuario. En esta se muestra la relación entre la capa de vista con la construcción de la página show.gsp por el controlador UsuarioController encargado además del manejo de peticiones de las vistas y la recepción de la información enviada por el formulario formUsuarioShow, así como la relación de uso que tiene el controlador con el servicio y este con la clase de dominio.

La leyenda muestra la definición de las clases en los DCD.

Las Vistas  Las Clases de Dominio  Los Controladores  Los Servicios

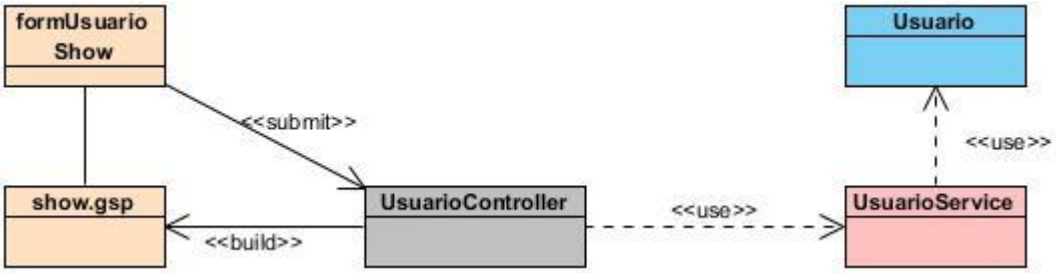


Figura 4: Diagrama de clases de diseño Ver Usuario.

## 4.2 Patrones de diseño.

Los patrones de diseño software son los que permiten describir fragmentos de diseño y reutilizar ideas de diseño, ayudando a beneficiarse de la experiencia de otros. Los patrones dan nombre y forma a heurísticas abstractas, reglas y buenas prácticas de técnicas orientadas a objetos. (66) El uso de estos patrones permitió resolver problemas de diseños. Estos tienen como características que se ha comprobado su efectividad resolviendo problemas similares en ocasiones anteriores y son reusables, lo que significa que son aplicables a diferentes problemas de diseño en distintas circunstancias

### 4.2.1 Inversión del control.

La inversión de control (IoC de *"Inversion of Control"*) es un patrón utilizado por el framework Grails, según el cual las dependencias de un componente no deben gestionarse desde el propio componente para que éste sólo contenga la lógica necesaria para hacer su trabajo. (67)

Grails define el uso de servicios para no cargar los controladores con la lógica de negocio, esta se debe realizar en los Grails Services. Al utilizar servicios en Grails, solo se escribe una variable con su nombre, sin tener en cuenta la instanciación del servicio y su configuración. Grails configura Spring para que gestione su ciclo de vida y sus dependencias. (67) De esta forma se mantienen los componentes lo más sencillo posible, los controladores solo contienen su lógica, dándole más sencillez al código de la aplicación y liberando al programador de la carga de las configuraciones que conlleva.

### 4.2.2 Patrones GRASP.

Estos patrones permiten una correcta asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns" o Patrones Generales de Software para Asignar

Responsabilidades. El uso de estos patrones permitió la asignación de responsabilidades y desglose de los artefactos de la aplicación por sus funciones (66). A continuación se muestran los patrones de responsabilidad que se implementan:

**Bajo acoplamiento**: El patrón propone el diseño de clases que son más independientes, lo que reduce el impacto del cambio y facilita la reutilización en otros sistemas. (66) Fue utilizado en las clases de dominio debido a que estas manejan solo la información relacionada con ellas y permite separarla de las demás.

**Alta cohesión**: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. (66) Un ejemplo de su utilización son las clases de dominio que implementan solo el negocio que le corresponde. La figura 9 muestra el desglose de la clase Usuario y sus componentes. La cohesión y el acoplamiento están fuertemente relacionados y, normalmente, una mala cohesión causa un mal acoplamiento, y viceversa.

**Experto**: Asigna la responsabilidad de realizar una tarea determinada, a aquel objeto que tiene la información necesaria para ello. (66) Este patrón se evidencia en la asignación de la responsabilidad de la creación de objetos por parte de los controladores en los servicios.

**Creador**: Este patrón como su nombre lo indica es el que crea, el que guía la asignación de responsabilidades relacionadas con la creación de objetos. (66) Este patrón se evidencia en los servicios de grails, responsables de manejar la lógica del negocio relacionado con los datos de la clase de dominio que maneja.

**Controlador**: Se encarga de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. El objeto controlador no será el que realice estas actividades, sino que las delegará en otras clases con las que mantiene un modelo de alta cohesión. (66) Este patrón se refleja en los controladores de grails (**ver figura 10**). Estos obtienen la información proveniente de las vistas y son capacees de decidir qué hacer con esta información, esto incluye la comunicación con la capa modelo para que se realice una determinada función sobre el negocio que se maneja.

### 4.2.3 Patrones GOF.

**Singleton:** garantiza que exista un único objeto que almacena la información del usuario que trabaja con la aplicación. Grails controla el ciclo de vida de los servicios, por defecto, estos servicios son Singleton

(67). Se utiliza en la instanciación de los servicios en los controladores (**ver figura 8**). Esto da la posibilidad de usar el mismo servicio para todo el sistema reduciendo la creación de instancias del mismo, mejorando el rendimiento de la aplicación.

### 4.3 Tarjetas Clase – Responsabilidad – Colaborador.

La metodología ágil XP constituye un modo simple de organizar las clases más notables para las funcionalidades del sistema. Estos modelos son las tarjetas CRC (Clase – Responsabilidad – Colaborador) de esta manera organiza las clases más relevantes de forma simple.

Las tarjetas CRC se dividen en tres secciones: el nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda y constituye lo que la clase sabe o hace, y por último a la derecha colaboradores que implica otras clases con las que trabaja en conjunto para realizar las funcionalidades.

En la siguiente tabla se muestra la tarjeta CRC correspondiente a la clase Caso. A continuación se muestran las tarjetas CRC correspondientes a las clases más importantes definidas:

| **Clase: Caso** | |
|---|---|
| Responsabilidad | **Colaborador** |
| Permite insertar y modificar los rasgos predictores.<br>Permite insertar y modificar el rasgo objetivo.<br>Permite insertar, modificar, ver y eliminar un caso. | Caso_Controller |

Tabla 12: Tarjeta CRC Caso.

| **Clase: Relevancia** | |
|---|---|
| Responsabilidad | **Colaborador** |
| Permite insertar, modificar, ver y eliminar una relevancia | Relevancia_Controller |

Tabla 13: Tarjeta CRC Relevancia.

| **Clase: Categoría** | |
|---|---|
| Responsabilidad | **Colaborador** |
| Permite insertar, modificar, ver y eliminar una categoría. | Categoria_Controller |

Tabla 14: Tarjeta CRC Categoría.

### 4.4 Modelo Físico de la Base de Datos.

Para un mejor entendimiento de lo descrito en las tarjetas CRC se muestra el diseño del modelo físico de la base de datos del sistema. En este se define la tabla baseDeCasos relacionada de uno a mucho con caso, este contienen una relación de muchos con rasgoPredictor y de uno a uno con rasgoObjetivo, el rasgoPredictor a su vez tiene una relación de muchos a uno con categoría y relevancia, la tabla usuario tiene una relación e muchos a muchos con rol y de uno a muchos con caso. Estas relaciones se muestran en la siguiente figura:

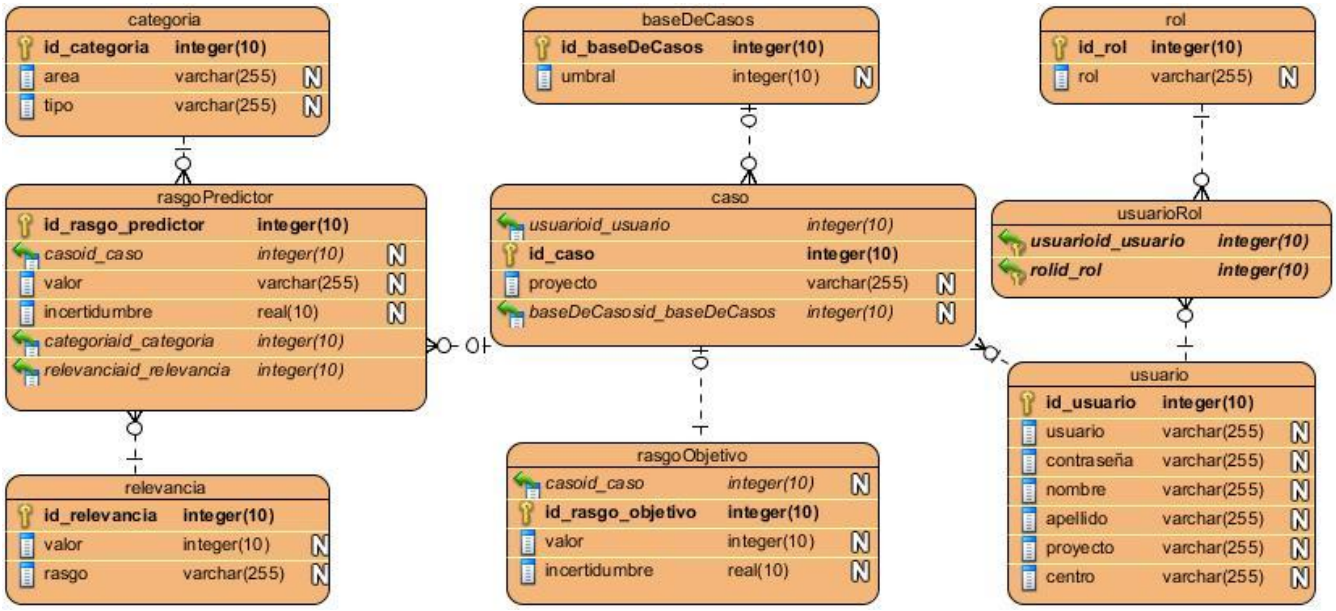


Figura 5: Diagrama del modelo físico de la base de datos.

### 4.5 Tareas de Ingeniería.

Las tareas de ingeniería partirán de las HU y serán implementadas por los desarrolladores en sus respectivas iteraciones, estas tareas serán escritas en leguaje técnico. A continuación se muestran las tareas de ingeniería en las iteraciones correspondientes:

**Iteración 1.**

El principal objetivo de estas iteraciones desarrollar las HU1.Autenticar usuario, 2.Gestionar usuario, 3. Gestionar Rol y 4. Gestionar relación usuario-rol. Solo se mostrará la tarea 1 de la HU 1 para ver las tareas de la HU 2, 3 y 4 remitirse **Anexo 21**. Para ello se definieron las siguientes tareas:

- Tarea Nº 1: Autenticar usuario
- Tarea Nº 2: Insertar usuario
- Tarea Nº 3: Ver usuario
- Tarea Nº 4: Modificar usuario
- Tarea Nº 5: Eliminar usuario
- Tarea Nº 6: Listar usuario
- Tarea N$^0$ 7: Insertar rol
- Tarea Nº 8: Ver rol
- Tarea Nº 9: Modificar rol
- Tarea Nº 10: Eliminar rol
- Tarea Nº 11: Listar rol
- Tarea N$^0$ 12: Insertar usuario-rol
- Tarea Nº 13: Ver usuario-rol
- Tarea Nº 14: Modificar usuario-rol
- Tarea Nº 15: Eliminar usuario-rol
- Tarea Nº 16: Listar usuario-rol

| **Tarea** | |
|---|---|
| **Número de tarea**: 1 | **Número de HU**: 1 |
| **Nombre de la tarea:** Autenticar usuario | |
| **Tipo de tarea**: Desarrollo | **Estimación**: 1/7 |
| **Programador responsable**: Pedro Carlos Abreu Jiménez y Jeny Del Sol Chang | |
| **Descripción:** Permite la entrada de los usuarios al sistema y dependiendo del tipo de usuario que sea y los privilegios que este pueda tener, se muestra la interfaz correspondiente. | |

Tabla 15: Tarea: Autenticar usuario.

**Iteración 2.**

El principal objetivo de estas iteraciones desarrollar las HU 5.Gestionar caso, y HU 6.Gestionar umbral. Solo se mostrará la tarea 17 de la HU.5 para ver las tareas de la HU 5 y la HU 6 remitirse al **Anexo 22**. Para ello se definieron las siguientes tareas:

- Tarea Nº 17: Insertar caso
- Tarea Nº 18: Ver caso
- Tarea Nº 19: Modificar caso
- Tarea Nº 20: Listar caso
- Tarea Nº 21: Insertar umbral

- Tarea Nº 22: Modificar umbral

| **Tarea** | |
|---|---|
| **Número de tarea**: 17 | **Número de HU**: 5 |
| **Nombre de la tarea:** Insertar caso | |
| **Tipo de tarea**: Desarrollo | **Estimación**: 2/7 |
| **Programador responsable**: Pedro Carlos Abreu Jiménez y Jeny Del Sol Chang | |
| **Descripción:** Se crea el caso después de insertar los datos correspondientes. | |

Tabla 16: Tarea**:** Insertar caso.

**Iteración 3.**

El principal objetivo de esta iteración es desarrollar las HU 7.Gestionar relevancia de los rasgos y HU 8.Gestionar categoría de los rasgos. Solo se mostraran la tarea 24 referente a la HU 7 para ver las demás tareas de la HU 7 y la HU 8 remitirse al **Anexo 23**. Para ello se definieron las siguientes tareas:

- Tarea Nº 23: Insertar relevancia
- Tarea Nº 24: Ver relevancia
- Tarea Nº 25: Modificar relevancia
- Tarea Nº 26: Eliminar relevancia
- Tarea Nº 27: Insertar categoría
- Tarea Nº 28: Ver categoría
- Tarea Nº 29: Modificar categoría
- Tarea Nº 30: Eliminar categoría
- Tarea Nº 31: Listar categoría

| **Tarea** | |
|---|---|
| **Número de tarea**: 23 | **Número de HU**: 7 |
| **Nombre de la tarea:** Insertar relevancia | |
| **Tipo de tarea**: Desarrollo | **Estimación**: 2/7 |
| **Programador responsable**: Pedro Carlos Abreu Jiménez y Jeny Del Sol Chang | |
| **Descripción:** Se crea el dato correspondiente a la relevancia. | |

Tabla 17: Tarea**:** Insertar relevancia.

**4.6 Pruebas.**

En la metodología XP las pruebas son importantes ya que mediante las mismas se puede comprobar tanto la eficiencia del software como que los objetivos planeados en la etapa de implementación se hayan

cumplido garantizando de esta manera que aumente la seguridad y disminuyan los errores del sistema. (64)

La estrategia de prueba seguida para la realización de las mismas sobre el sistema contempla los niveles de prueba: prueba de unidad, prueba de sistema y pruebas de aceptación. Para la realización de las pruebas de unidad se realizaron pruebas automáticas haciendo uso del sistema de pruebas que brinda el propio Framework utilizado. Las pruebas de unidad están orientadas a caja blanca. La realización de las pruebas de sistema y de aceptación se efectuó de forma manual donde incluye el diseño de casos de prueba utilizando el método de caja negra.

Prueba de caja negra: Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software donde los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. (68)

**Pruebas de unidad.**

Estas pruebas se centran principalmente en los requisitos funcionales del software. Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (24)

El framework utilizado brinda su propio sistema de pruebas (test). Los test unitarios son aquellos en los que se verifica que un metodo se comporta como deberia, sin tener en cuenta su entorno. Cuando se ejecutan las pruebas unitarias de un método, Grails no inyecta ninguno de los métodos dinámicos con los que se cuentan en la aplicación que se está ejecutando. Siendo los desarrolladores los responsables de crear y gestionar todos los objetos. (67) Para ejecutar las consulta dinámicas se utilizan métodos definidos en el entorno. Esta pruebas se realizan mientra se desarrolla y se permite visualizar los resultados de los test en un archivo HTML. El resultado de las pruebas se puede apreciar en el **Anexo 24** y el **Anexo 25,** en estos se muestra el resultado de las mismas**.**

**Pruebas de sistema.**

Las pruebas de caja negra se centran principalmente en ejercitar los requisitos funcionales del sistema mediante un conjunto de condiciones de entrada, por lo que se utilizó las pruebas de sistema. Estas pruebas se centran en verificar la aplicación (y sus procesos internos) mediante la interacción con la interfaz del sistema. Para la ejecución de estas pruebas, usando datos válidos e inválidos, se pretende verificar que los resultados esperados ocurran cuando se usen datos válidos y que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos. (68) Se tomó una pequeña muestra de los indicadores que conforman el caso para plasmarlo en el documento debido a la gran cantidad de indicadores que se validaron, este caso de prueba se pueden observar en el **Anexo 26**, las no conformidades encontradas por iteración se corrigieron quedando como resultado en el **Anexo 27**.

**Pruebas de aceptación**

Las pruebas de aceptación son consideradas como ¨pruebas de caja negra¨, son creadas en base a las historias de usuario, dichas historias no se pueden considerar terminadas hasta tanto pase correctamente por las pruebas de aceptación. Se debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. (64) Estas pruebas se realizaron con el objetivo de verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. Se conformaron las pruebas aceptación basados en las pruebas de caja negra, utilizando los casos de pruebas definidos en el **Anexo 26**. Estas pruebas resultaron satisfactorias.

**Conclusiones.**

En este capítulo se abordó la etapa de diseño, implementación y prueba de la Base de Casos para estimar la duración de proyectos de desarrollo de software. Se describió la arquitectura Modelo-Vista-Controlador que utiliza el framework Grails, se identificaron los patrones de diseño, se describieron las tareas de ingeniería necesarias por cada HU para dar cumplimiento a los requisitos establecidos en el tiempo requerido. Se definieron y realizaron las pruebas al sistema.

**CONCLUSIONES**

Con el desarrollo del presente trabajo se arribó a las siguientes conclusiones:

- Se realizó un estudio de los métodos de estimación Puntos por Casos de Uso, Puntos de Función, COCOMO, PROBE y el método de estimación UCI y se definieron los indicadores relacionados con la estimación de la duración de proyectos de desarrollo de software.
- Se formalizó un modelo computacional donde se aplican técnicas planteadas por los sistemas basados en conocimiento para estimar la duración de proyectos de desarrollo de software basados en los resultados obtenidos de la selección de indicadores.
- Obtenido el diseño de la base de casos se realizó el análisis y diseño del sistema.
- Obtenido el análisis y diseño del sistema se implementó la solución propuesta.
- Se realizaron las correspondientes validaciones a la solución a través de las pruebas al sistema.

**RECOMENDACIONES**

Realizado el desarrollo del presente trabajo, se recomiendan que:

- Se provea de un mecanismo para incorporar al sistema los módulos de recuperación, adaptación y aprendizaje para poder ser usado en la estimación de la duración de proyectos en la Universidad de la Ciencias Informáticas.
- Se propone el cálculo del umbral de semejanza que se basa en la información almacenada en la base de casos.